2e édition

Christian Tavernier

Les microcontrôleurs PIC

Description et mise en œuvre

DUNOD

Collection EEA
Série Technologie électronique

Christian Tavernier

LES MICROCONTRÔLEURS PIC
Description et mise en œuvre

Inconnus il y a une dizaine d'années, les microcontrôleurs PIC de Microchip jouissent aujourd'hui d'un succès planétaire dû principalement à leur rapport coût/performances particulièrement attractif.

Cette deuxième édition de *Les microcontrôleurs PIC - Description et mise en œuvre*, qui est consacrée aux deux familles principales de ce composant — PIC 12Cxxx et PIC 16Cxxx —, a été entièrement repensée de sorte que le lecteur possède le premier livre à jour en permanence grâce à Internet. Tout d'abord, chaque constituant de base des microcontrôleurs PIC (UC, E/S parallèles, E/S séries, etc.) est décrit dans un chapitre dédié. Des tableaux de sélection pouvant être mis à jour via le [illegible] d'établir la liste des constituants d'un circuit donné, il reste alors à se reporter aux chapitres correspondants pour « reconstituer » la fiche technique complète du circuit choisi.

Accompagné de son double cédérom, *Les microcontrôleurs PIC - Description et mise en œuvre* deviendra vite le compagnon indispensable à toute personne désireuse d'obtenir le maximum de ce merveilleux composant.

1. Présentation générale. 2. Horloge et *reset*. 3. Architecture interne. 4. Mémoire EEPROM de données. 5. *Timer* chien de garde ou WDT et mode sommeil. 6. Ports d'entrées/sorties parallèles. 7. *Timers*. 8. Modules CCP et PWM. 9. Port série synchrone ou SSP. 10. Interface de communication série ou SCI. 11. Source de tension de référence et comparateurs. 12. Convertisseurs analogiques digitaux 8 et 10 bits. 13. Interface pour afficheur à cristaux liquides. 14. Programmation en circuit ou ICSP. 15. Adressage et jeu d'instructions.

ISBN 2 10 005266 7
Code 045266

http://www.dunod.com

PRINCIPES ÉLECTRONIQUES
TECHNOLOGIE ÉLECTRONIQUE
ÉLECTROTECHNIQUE
AUTOMATISMES

2e édition

CHRISTIAN TAVERNIER
Ingénieur conseil en électronique, Christian Tavernier est aussi, et surtout, un spécialiste incontesté des microprocesseurs et microcontrôleurs depuis plus de vingt ans, dont les nombreux articles et ouvrages de vulgarisation font autorité.

2 CD-ROM inclus

DUNOD

C. TAVERNIER
LES MICROCONTRÔLEURS PIC
DESCRIPTION ET MISE EN ŒUVRE

INF
712

2e édition
Christian Tavernier
Les microcontrôleurs PIC
Description et mise en œuvre
2 CD-ROM inclus
DUNOD

INF712

Technologie électronique

Christian Tavernier

Les microcontrôleurs PIC

Description et mise en œuvre

2e édition

DUNOD

INF 712

L'auteur tient à remercier la société Microchip France d'avoir accepté l'encartage — à titre gracieux — de son dernier cédérom dans cet ouvrage.

Ce pictogramme mérite une explication. Son objet est d'alerter le lecteur sur la menace que représente pour l'avenir de l'écrit, particulièrement dans le domaine de l'édition technique et universitaire, le développement massif du **photocopillage**.

Le Code de la propriété intellectuelle du 1er juillet 1992 interdit en effet expressément la photocopie à usage collectif sans autorisation des ayants droit. Or, cette pratique s'est généralisée dans les établissements d'enseignement supérieur, provoquant une baisse brutale des achats de livres et de revues, au point que la possibilité même pour les auteurs de créer des œuvres nouvelles et de les faire éditer correctement est aujourd'hui menacée.

Nous rappelons donc que toute reproduction, partielle ou totale, de la présente publication est interdite sans autorisation du Centre français d'exploitation du droit de copie (**CFC**, 20 rue des Grands-Augustins, 75006 Paris).

© Dunod, Paris, 2000
© Dunod, Paris, 1995 pour la première édition
ISBN 2 10 005266 7

Toute représentation ou reproduction intégrale ou partielle faite sans le consentement de l'auteur ou de ses ayants droit ou ayants cause est illicite selon le Code de la propriété intellectuelle (Art L 122-4) et constitue une contrefaçon réprimée par le Code pénal. • Seules sont autorisées (Art L 122-5) les copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à une utilisation collective, ainsi que les analyses et courtes citations justifiées par le caractère critique, pédagogique ou d'information de l'œuvre à laquelle elles sont incorporées, sous réserve, toutefois, du respect des dispositions des articles L 122-10 à L 122-12 du même Code, relatives à la reproduction par reprographie.

© Dunod. La photocopie non autorisée est un délit.

TABLE DES MATIÈRES

AVANT-PROPOS 1

CHAPITRE 1 - PRÉSENTATION GÉNÉRALE 3

1.1 Harvard contre Von Neumann 3
1.2 Les points fort de l'architecture RISC 4
1.3 Les différentes familles de PIC 5
1.4 Dans la jungle des références 11
1.5 Les différentes mémoires de programme 11
1.6 Tensions d'alimentation 13
1.7 Appellations normalisées des diverses pattes disponibles 14

CHAPITRE 2 - HORLOGE ET *RESET* 19

2.1 Les horloges des PIC 19
2.1.1 Les différents types d'horloges 19
2.1.2 Les schémas d'horloges types 21
2.1.3 L'horloge RC interne et son registre de calibration 23
2.1.4 Comment récupérer le contenu de OSCCAL ? 25
2.2 Le *reset* 26
2.2.1 Principes généraux 26
2.2.2 Circuiterie de *reset* externe 31
2.2.3 Schéma interne de la circuiterie de *reset* 32
2.2.4 Le *reset* par détection de chute de tension d'alimentation ou BOR 35
2.2.5 Les registres de contrôle et d'état du *reset* 36

CHAPITRE 3 - ARCHITECTURE INTERNE 39

3.1 Architecture interne 39
3.2 Organisation de la mémoire 41
3.2.1 Mécanisme de pagination de la mémoire de programme 42
3.2.2 Mécanisme de pagination de la mémoire de données . 47
3.2.3 Organisation de la mémoire de données et des registres internes 48
3.3 Les registres de l'unité centrale 50
3.3.1 Le registre INDF d'adresse 00 ou registre d'indirection . 50
3.3.2 Le registre PCL d'adresse 02. 50
3.3.3 Le registre STATUS d'adresse 03 ou registre d'état. . . . 51
3.3.4 Le registre FSR d'adresse 04 ou registre de sélection de registre. 53
3.3.5 Le registre PCLATH d'adresse 0A. 53
3.3.6 Le registre PCON d'adresse 8E. 53
3.3.7 Le registre OPTION d'adresse 81 54
3.4 Les interruptions 56
3.4.1 Principe de fonctionnement des interruptions 57
3.4.2 Le registre INTCON d'adresse 0B 58
3.4.3 Les registres PIEx 59
3.4.4 Les registres PIRx 61

CHAPITRE 4 - LA MÉMOIRE EEPROM DE DONNÉES 63

CHAPITRE 5 - LE *TIMER* CHIEN DE GARDE OU WDT ET LE MODE SOMMEIL 67

5.1 Le *timer* chien de garde 67
5.2 Le mode sommeil 70
5.2.1 Particularités du mode sommeil 71
5.2.2 Le réveil ou la sortie du mode sommeil 71
5.2.3 Réveil au moyen des interruptions 72

CHAPITRE 6 - LES PORTS D'ENTRÉES/SORTIES PARALLÈLES 73

6.1 Généralités 73
6.2 Le port A 76
6.3 Le port B 78
6.4 Le port C 80
6.5 Le port D 81

6.6 Le port E 82
6.7 Les ports F et G 83
6.8 Le port GPIO 84
6.9 Le mode port parallèle esclave ou PSP 86

CHAPITRE 7 - LES *timers* 91

7.1 Le *timer 0* 91
7.2 Le *timer 1* 99
7.3 Le *timer 2* 101

CHAPITRE 8 - LES MODULES DE CAPTURE ET COMPARAISON (CCP) ET DE MODULATION DE LARGEUR D'IMPULSIONS (PWM) 105

8.1 Le mode capture 107
8.2 Le mode comparaison 108
8.3 Le mode modulation de largeur d'impulsions ou mode PWM . . 109
8.4 Le registre de contrôle des modules CCP 111

CHAPITRE 9 - LE PORT SÉRIE SYNCHRONE OU SSP 113

9.1 Le SSP en mode SPI 113
9.2 Le SSP en mode I2C 118
9.3 Les registres du SSP 127

CHAPITRE 10 - L'INTERFACE DE COMMUNICATION SÉRIE OU SCI 131

10.1 La SCI en mode asynchrone *full duplex* 131
10.2 La SCI en mode synchrone *half duplex* 134
10.3 Les registres de la SCI 135

CHAPITRE 11 - LA SOURCE DE TENSION DE RÉFÉRENCE ET LES COMPARATEURS 141

11.1 La source de tension de référence 141
11.2 Les comparateurs de tension 143

© Dunod – Toute reproduction non autorisée est un délit.

CHAPITRE 12 - LES CONVERTISSEURS ANALOGIQUES/DIGITAUX 8 ET 10 BITS 149

12.1 Le convertisseur analogique/digital 8 bits 149
12.1.1 Utilisation du convertisseur en mode SLEEP. 156
12.1.2 Utilisation du convertisseur avec un module CCP. 156
12.2 Le convertisseur analogique/digital 10 bits 157

CHAPITRE 13 - L'INTERFACE POUR AFFICHEURS À CRISTAUX LIQUIDES . . 163

13.1 Principe général 163
13.2 Horloges et multiplexage 168
13.3 Génération des tensions de l'afficheur 171
13.4 Utilisation de l'interface LCD 173

CHAPITRE 14 - LA PROGRAMMATION EN CIRCUIT OU ICSP 179

14.1 Mise en mode programmation 179
14.2 Contraintes de la programmation en circuit 180

CHAPITRE 15 - ADRESSAGE ET JEU D'INSTRUCTIONS 183

15.1 Les modes d'adressage 183
15.2 Le jeu d'instructions 185
15.3 Compatibilité ascendante 187
15.4 Description détaillée des instructions 188
15.5 Tableaux synthétiques 200

CONCLUSION 203

ANNEXE - CÉDÉROM MICROCHIP. 205

INDEX 207

© Dunod. La photocopie non autorisée est un délit.

AVANT-PROPOS

Totalement inconnus il y a une dizaine d'années, les microcontrôleurs PIC de Microchip connaissent aujourd'hui un succès planétaire et sont omniprésents dans les appareils les plus divers.

La raison d'être de ce succès tient tout à la fois au coût particulièrement bas de ces circuits et à leurs performances remarquables. Leur jeu d'instructions réduit, plus connu sous l'acronyme de RISC (*reduced instruction set computer*), et leur architecture interne de type Harvard y sont pour beaucoup.

Ces deux approches déroutent un peu lors de la lecture des fiches techniques de ces circuits et peuvent faire croire, à tort, qu'ils sont d'abord plus difficiles que des microcontrôleurs « classiques ». Cet ouvrage est là pour vous démontrer le contraire.

Nous vous proposons en effet de découvrir les deux familles principales que sont les PIC 12Cxxx et les PIC 16Cxxx et, pour que ce livre soit d'utilisation aussi pratique que possible, nous l'avons organisé d'une façon un peu particulière.

En effet, comme chez Microchip la notion de famille n'est pas un vain mot, tous les circuits sont réalisés par assemblage des mêmes blocs fonctionnels dont la présence, le nombre ou l'absence différencient les divers circuits proposés sur le marché.

Après une présentation générale des notions relatives à ces circuits au chapitre 1, vous y découvrirez donc des tableaux indiquant le contenu de chaque circuit, référence par référence.

Chaque chapitre suivant de l'ouvrage est ensuite consacré à la présentation d'une ou plusieurs des ressources internes mentionnées dans ces tableaux, pour terminer par le jeu d'instruction complet.

Ainsi, en présence de n'importe quel type de circuit PIC des familles 12Cxxx ou 16Cxxx, il vous suffira de vous reporter au chapitre 1 pour savoir « ce qu'il contient » et de lire ensuite les chapitres correspondants pour reconstituer la fiche technique complète de votre circuit.

CHAPITRE 1

PRÉSENTATION GÉNÉRALE

Commercialisés pour la première fois il y a seulement dix ans, les microcontrôleurs PIC auraient pu n'avoir qu'un succès mitigé car ils arrivaient sur un marché déjà bien encombré, et encombré par des « poids lourds » du métier de surcroît tels que Motorola ou Intel. Malgré cela, leur succès fut immédiat et fulgurant et l'on peut affirmer aujourd'hui que ces circuits sont devenus de véritables références dans le monde des microcontrôleurs. Ce succès, ils le doivent entre autres choses à deux caractéristiques particulières de leur architecture interne dont nous allons maintenant dire quelques mots en guise d'introduction à cette présentation générale.

1.1 Harvard contre Von Neumann

La majorité des microprocesseurs et microcontrôleurs actuels utilise une architecture interne dite de Von Neumann, c'est-à-dire en fait une architecture commune à celle que l'on rencontre habituellement dans les micro-ordinateurs. La mémoire, appelée improprement « de programme », contient en fait des instructions et des données placées les unes à la suite des autres et l'on ne dispose que d'un bus, appelé bus de données, pour véhiculer tour à tour les codes des instructions et les données qui leur sont associées, comme le montre la *figure 1.1.*

Si cette architecture donne toute satisfaction comme vous en avez la preuve éclatante chaque jour, elle pose quelques problèmes dès que l'on veut faire fonctionner l'ensemble rapidement. En effet, l'exécution d'une seule instruction nécessite plusieurs échanges de données sur le seul et unique bus dévolu à cet usage puisqu'il faut tout d'abord aller chercher le code de l'instruction, puis le ou les données qu'elle doit manipuler.

Il est alors préférable de faire appel à une architecture dite Harvard dans laquelle les instructions et les données sont clairement différenciées et sont véhiculées sur des bus différents comme le montre la *figure 1.2.* Vu de l'utilisateur, cela ne

change rien bien sûr et les circuits de ce type s'utilisent exactement comme les autres. Par contre, les résultats obtenus, en termes de vitesse d'exécution des programmes, peuvent être impressionnants.

En effet, l'exécution d'une instruction ne fait plus appel qu'à un seul cycle machine puisque l'on peut simultanément, grâce aux deux bus, rechercher le code de l'instruction et la ou les données qu'elle manipule.

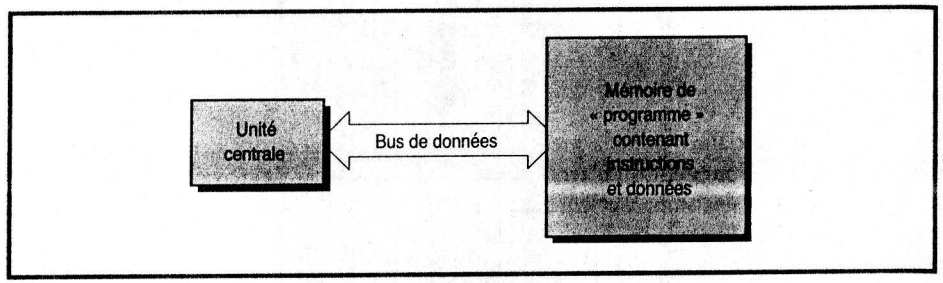


Figure 1.1 – Principe de l'architecture de Von Neumann.

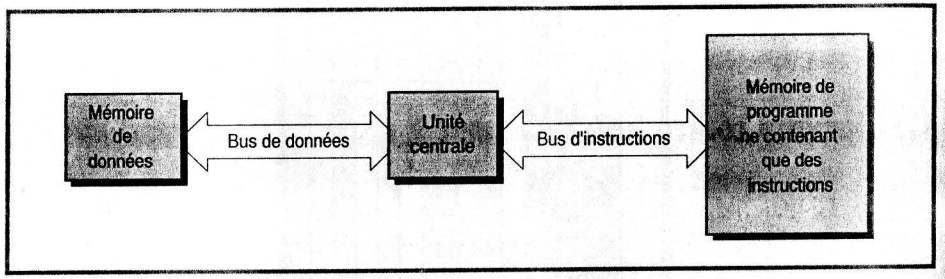


Figure 1.2 – Principe de l'architecture Harvard.

Rompant avec une tradition bien établie, les microcontrôleurs PIC de Microchip utilisent une architecture Harvard mais ce n'est pas tout. Ils font également appel à une architecture de type RISC qui, comme on le lit un peu trop souvent, ne se résume pas simplement à disposer d'un jeu d'instructions réduit.

1.2 Les points fort de l'architecture RISC

RISC signifie *reduced instruction set computer*, ce qui veut effectivement dire circuit à jeu d'instructions réduit mais ne se limite pas uniquement à ça. Un vrai

circuit de type RISC doit en effet présenter un certain nombre de particularités propres à accroître sa vitesse de fonctionnement.

Les microcontrôleurs à architecture RISC utilisent des instructions codées sur un seul mot (de 12 ou 14 bits de long dans le cas des PIC), ce qui présente deux avantages. Le premier est que tous les emplacements de la mémoire de programme contiennent une instruction, le second est qu'un seul cycle machine suffit pour lire le code complet d'une instruction, d'où un gain en vitesse d'exécution.

Les circuits RISC utilisent ensuite une structure de type pipe-line qui leur permet d'exécuter une instruction tout en recherchant la suivante en mémoire d'où, là encore, accroissement de la vitesse d'exécution.

Les circuits RISC exécutent toutes leurs instructions en un seul cycle machine, ce qui est dû en grande partie au codage de l'instruction sur un seul mot. L'ALU (unité arithmétique et logique) dispose en effet en une seule fois de toutes les informations nécessaires à l'exécution de l'instruction.

Les instructions d'un circuit de type RISC ont une structure aussi orthogonale que possible, c'est-à-dire que toutes les instructions peuvent être exécutées sur tous les registres avec tous les modes d'adressage. Cela simplifie le travail du programmeur, et donc le vôtre, car il n'est plus nécessaire de retenir une multitude de cas particuliers d'instructions qui ne fonctionnent que dans un mode ou dans un autre.

Enfin, et c'est hélas la seule chose que l'on retient trop souvent à propos de l'architecture RISC, vous n'avez plus à apprendre une centaine d'instructions différentes comme avec de nombreux microcontrôleurs actuels mais « seulement » 33 ou 35 instructions (dans le cas des PIC) que vous découvrirez dans la suite de cet ouvrage.

1.3 Les différentes familles de PIC

Il existe aujourd'hui cinq grandes familles de microcontrôleurs PIC qui se répartissent comme suit.

La famille 12Cxxx est d'introduction sur le marché relativement récente. Elle est constituée de circuits contenus en boîtiers 8 pattes, ce qui la destine aux plus petites applications à base de microcontrôleurs. Du fait de ces boîtiers de taille réduite, les circuits sont caractérisés par un niveau d'intégration particulièrement poussé qui leur permet, si nécessaire, de se passer de tout composant externe pour leur circuit d'horloge et de *reset*. Ces circuits utilisent en outre une unité arithmétique et logique travaillant avec des mots de 12 bits (14 bits pour les versions les plus récentes) et ne disposent « que » de 33 instructions (35 pour les versions les

© Dunod. La photocopie non autorisée est un délit.

plus récentes) comme nous le verrons au moment opportun. Malgré cela, ils rencontrent actuellement un très grand succès car leur petite taille, leur facilité de mise en œuvre et leur très faible coût permet de les utiliser dans des applications où, par le passé, personne n'aurait envisagé de mettre un microcontrôleur.

La deuxième famille est, historiquement, celle qui a contribué à lancer Microchip dans le monde des microprocesseurs puisque c'est la famille 16C5x. C'est la famille la plus ancienne, et ces circuits se rencontrent aujourd'hui encore dans de nombreuses applications économiques. Cependant, compte tenu de leur âge et des limitations technologiques qui en découlent, les produits de cette famille sont en voie de disparition avec l'arrivée, dans la famille 16Cxxx, de composants plus performants qui ne sont pas plus coûteux. De plus, la programmation des circuits de cette famille est relativement complexe alors que tous les autres microcontrôleurs Microchip se programment facilement, en circuit si c'est nécessaire, et avec seulement cinq fils.

La famille la plus connue à l'heure actuelle est la famille 16Cxxx, qualifiée initialement par Microchip de famille *mid-range*, ce que l'on peut traduire par « milieu de gamme ». C'est elle qui comprend actuellement le plus de références et qui se développe le plus rapidement. Ces circuits reprennent évidemment les concepts qui ont fait le succès de la famille 16C5x. Ils utilisent la même architecture et le même jeu d'instructions de base mais disposent de ressources internes plus nombreuses avec, par exemple, des ports séries variés, des ports parallèles plus nombreux, ou bien encore des convertisseurs analogiques/digitaux. Des possibilités supplémentaires existent aussi au niveau de la gestion des interruptions.

Enfin, deux autres familles doivent être signalées bien qu'elles ne soient pas traitées dans cet ouvrage, encore qu'elles présentent une compatibilité ascendante avec les circuits présentés ici. Ce sont les circuits 17Cxxx et 18Cxxx qui sont actuellement les plus « gros » microcontrôleurs de chez Microchip et qui, sans être absents du marché, sont loin de rencontrer le succès des familles 12Cxxx et 16Cxxx.

Le contenu de ces différentes familles est résumé dans les *tableaux 1.1*, *1.2* et *1.3* mais, compte tenu de l'évolution rapide (trop rapide même diront certains utilisateurs) des microcontrôleurs PIC, et bien que ces tableaux soient à jour à la date de publication de cette édition, nous ne saurions trop vous recommander, avant de décider de réaliser une application, de consulter les dernières informations en vigueur chez Microchip. Vous pouvez pour cela vous procurer, auprès d'un distributeur de la marque, le dernier data book sur cédérom ou bien encore vous connecter au site Internet de Microchip, dont l'adresse est tout simplement : *www.microchip.com*. Ce site, fort bien fait, contient en effet, entre autres choses, toutes les fiches techniques et notes d'applications relatives aux microcontrôleurs PIC.

© Dunod. La photocopie non autorisée est un délit.

Tableau 1.1 – Caractéristiques résumées des circuits de la famille PIC 12CXXX.

Référence	Mémoire de programme			Mémoire RAM	Analogique			Numérique				Système	Autres particularités	Boîtiers
	Octets	Taille OTP/Flash	Taille ROM	Taille RAM	Canaux CAN (bits)	Comparateurs	Ports d'E/S	E/S série	PWM 10 bits	Timers	Vitesse (MHz)			
PIC12CXXX – 400 ns Instruction Execution, 33.35 Instructions, 8 Pin Package, 4 MHz Internal Oscillator, 4:5 Oscillator Selections														
PIC12C508	768	512x12	–	25	–	–	6	–	–	1-8bit, 1-WDT	4	Yes	25mA source/sink per I/O	8P, RSM, SJW
PIC12C508A	768	512x12	–	25	–	–	6	–	–	1-8bit, 1-WDT	4	Yes	25mA source/sink per I/O	8P, RSM, SJW
PIC12C509	1536	1024x12	–	41	–	–	6	–	–	1-8bit, 1-WDT	4	Yes	25mA source/sink per I/O	8P, RSM, SJW
PIC12C509A	1536	1024x12	–	41	–	–	6	–	–	1-8bit, 1-WDT	4	Yes	25mA source/sink per I/O	8P, RSM, SJW, SN
PIC12CE518	768	512x12	–	25	–	–	6	–	–	1-8bit, 1-WDT	4	Yes	25mA source/sink per I/O	8P, RSM, SJW, RSN
PIC12CE519	1536	1024x12	–	41	–	–	6	–	–	1-8bit, 1-WDT	4	Yes	25mA source/sink per I/O	8P, RSM, SJW, RSN
PIC12C671	1794	1024x14	–	128	4	–	6	–	–	1-8bit, 1-WDT	10	Yes	25mA source/sink per I/O	8P, RSM, SJW
PIC12C672	3584	2048x14	–	128	4	–	6	–	–	1-8bit, 1-WDT	10	Yes	25mA source/sink per I/O	8P, RSM, SJW
PIC12CE673	1794	1024x14	–	128	4	–	6	–	–	1-8bit, 1-WDT	10	Yes	25mA source/sink per I/O	8P, SJW
PIC12CE674	3584	2048x14	–	128	4	–	6	–	–	1-8bit, 1-WDT	10	Yes	25mA source/sink per I/O	8P, SJW

Tableau 1.2 – Caractéristiques résumées des circuits de la famille PIC 16C5X.

Référence	Mémoire de programme			Taille EEPROM de données	Taille RAM	Analogique			Numérique				Système		Autres particularités	
	Octets	Taille OTP/Flash	Taille ROM			Canaux ADC 8 bits	Comparateurs	Autres	Ports d'E/S	E/S série	PWM 10 bits	Timers	Vitesse max en MHz	ICSP		
PIC16C5X – 200 ns Instruction Execution, 33 Instructions, 4/5 Oscillator Selections																
PIC16C52	576	384x12	–	–	25	–	–	–	12	–	–	1-8bit.	4	–	10mA source/sink per I/O	18P, 18SO
PIC16C54	768	512x12	–	–	25	–	–	–	12	–	–	1-8bit. 1-WDT	20	–	20mA source and 25mA sink per I/O	18P, 18JW, 18SO, 20SS
PIC16CR54A	768	–	512x12	–	25	–	–	–	12	–	–	1-8bit. 1-WDT	20	–	20mA source and 25mA sink per I/O	18P, 18SO, 20SS
PIC16C54A	768	512x12	–	–	25	–	–	–	12	–	–	1-8bit. 1-WDT	20	–	20mA source and 25mA sink per I/O	18P, 18JW, 18SO, 20SS
PIC16C54C	768	512x12	–	–	25	–	–	–	12	–	–	1-8bit. 1-WDT	20	–	20mA source and 25mA sink per I/O	18P, 18JW, 18SO, 20SS
PIC16CR54C	768	–	512x12	–	25	–	–	–	12	–	–	1-8bit. 1-WDT	20	–	20mA source and 25mA sink per I/O	18P, 18SO, 20SS
PIC16C55	768	512x12	–	–	24	–	–	–	20	–	–	1-8bit. 1-WDT	20	–	20mA source and 25mA sink per I/O	28P, 28JW, 28SP, 28SO, 28SS
PIC16C55A	768	512x12	–	–	24	–	–	–	20	–	–	1-8bit. 1-WDT	20	–	20mA source and 25mA sink per I/O	28P, 28JW, 28SP, 28SO, 28SS
PIC16C56	1536	1024x12	–	–	25	–	–	–	12	–	–	1-8bit. 1-WDT	20	–	20mA source and 25mA sink per I/O	18P, 18JW, 18SO, 20SS
PIC16C56A	1536	1024x12	–	–	25	–	–	–	12	–	–	1-8bit. 1-WDT	20	–	20mA source and 25mA sink per I/O	18P, 18JW, 18SO, 20SS
PIC16CR56A	1536	–	1024x12	–	25	–	–	–	12	–	–	1-8bit. 1-WDT	20	–	20mA source and 25mA sink per I/O	18P, 18SO, 20SS
PIC16C57	3072	2048x12	–	–	72	–	–	–	20	–	–	1-8bit. 1-WDT	20	–	20mA source and 25mA sink per I/O	28P, 28JW, 28SP, 28SO, 28SS
PIC16CR57B	3072	–	2048x12	–	72	–	–	–	20	–	–	1-8bit. 1-WDT	20	–	20mA source and 25mA sink per I/O	28P, 28SP, 28SO, 28SS
PIC16C57C	3072	2048x12	–	–	72	–	–	–	20	–	–	1-8bit. 1-WDT	20	–	20mA source and 25mA sink per I/O	28P, 28JW, 28SP, 28SO, 28SS
PIC16CR57C	3072	–	2048x12	–	72	–	–	–	20	–	–	1-8bit. 1-WDT	20	–	20mA source and 25mA sink per I/O	28P, 28SP, 28SO, 28SS
PIC16C58A	3072	2048x12	–	–	73	–	–	–	12	–	–	1-8bit. 1-WDT	20	–	20mA source and 25mA sink per I/O	18P, 18JW, 18SO, 20SS
PIC16CR58A	3072	–	2048x12	–	73	–	–	–	12	–	–	1-8bit. 1-WDT	20	–	20mA source and 25mA sink per I/O	18P, 18SO, 20SS
PIC16C58B	3072	2048x12	–	–	73	–	–	–	12	–	–	1-8bit. 1-WDT	20	–	20mA source and 25mA sink per I/O	18P, 18JW, 18SO, 20SS
PIC16CR58B	3072	–	2048x12	–	73	–	–	–	12	–	–	1-8bit. 1-WDT	20	–	20mA source and 25mA sink per I/O	18P, 18SO, 20SS
PIC16C505	1536	1024x12	–	–	72	–	–	–	12	–	–	1-8bit. 1-WDT	20	Yes	25mA source/sink per I/O 4MHz internal oscillator	14P, 14JW, 14SL
PIC16HV540*	768	512x12	–	–	25	–	–	BOD	12	–	–	1-8bit. 1-WDT	20	–	8 hight volatge (15V) I/Os, 4 deep stack, 5 I/O with wake up on change, Extended V_{DD} (3.5 to 15 V)	18P, 18JW, 18SO, 20SS

Tableau 1.3 – Caractéristiques résumées des circuits de la famille PIC 16CXXX.

Référence	Mémoire de programme			Taille EEPROM de données	Taille RAM	Analogique			Numérique				Système		Autres particularités	
	Octets	Taille OTP/Flash	Taille ROM			Canaux ADC 8 bits	Comparateurs	Autres	Ports d'E/S	E/S série	PWM 10 bits	Timers	Vitesse max en MHz	ICSP		
PIC16CXXX – 4-12 Interrupts, 200 ns Instruction Execution, 35 Instructions, 4/5 Oscillator Selections, Upwardly Compatible with PIC16C5X/PIC12CXXX																
PIC14C000	7168	4096x14	–	–	192	8 SLAC	2	VREF, Prog. Reference Generator	20	I2CTM/SMB	–	1-16 bit, 1-8 bit, 1-WDT	20	Yes	25mA source/sink Internal Oscillator Temperature Sensor	28SP, 28SO, 28SS, 28JW
PIC16C554	896	512x14	–	–	80	–	–	–	13	–	–	1-8 bit, 1-WDT	20	Yes	25mA source/sink per I/O	18SP, 18SO, 20SS, 18JW
PIC16C558	3584	2048x14	–	–	128	–	–	–	13	–	–	1-8 bit, 1-WDT	20	Yes	25mA source/sink per I/O	18SP, 18SO, 20SS, 18JW
PIC16C62A	3584	2048x14	–	–	128	–	–	BOD	22	I2C/SPITM	1	1-16 bit, 2-8 bit, 1-WDT	20	Yes	25mA source/sink per I/O, CCP	28SP, 28SO, 28SS, 28JW
PIC16CR62	3584	–	2048x14	–	128	–	–	BOD	22	I2C/SPI	1	1-16 bit, 2-8 bit, 1-WDT	20	–	25mA source/sink per I/O, CCP	28SP, 28SO, 28SS, 28JW
PIC16C62B	3584	2048x14	–	–	128	–	–	BOD	22	I2C/SPI	1	1-16 bit, 2-8 bit, 1-WDT	20	Yes	25mA source/sink per I/O, CCP	28SP, 28SO, 28SS, 28JW
PIC16C63	7168	4096x14	–	–	192	–	–	BOD	22	USART/I2C/SPI	2	1-16 bit, 2-8 bit, 1-WDT	20	Yes	25mA source/sink per I/O, 2 CCP	28SP, 28SO, 28JW
PIC16CR63	7168	–	4096x14	–	192	–	–	BOD	22	USART/I2C/SPI	2	1-16 bit, 2-8 bit, 1-WDT	20	Yes	25mA source/sink per I/O, 2 CCP	28SP, 28SO, 28SS, 28JW
PIC16C64A	3584	2048x14	–	–	128	–	–	BOD	33	I2C/SPI	1	1-16 bit, 2-8 bit, 1-WDT	20	–	25mA source/sink per I/O, 2 CCP	28SP, 28SO, 28SS
PIC16CR64	3584	–	2048x14	–	128	–	–	BOD	33	I2C/SPI	1	1-16 bit, 2-8 bit, 1-WDT	20	Yes	25mA source/sink per I/O, PSP, 2 CCP	40P, 40JW, 44L, 44PQ, 44PT
PIC16C65A	7168	4096x14	–	–	192	–	–	BOD	33	USART/I2C/SPI	2	1-16 bit, 2-8 bit, 1-WDT	20	Yes	25mA source/sink per I/O, PSP, 2 CCP	40P, 40JW, 44L, 44PQ, 44PT
PIC16C65B	7168	4096x14	–	–	192	–	–	BOD	33	USART/I2C/SPI	2	1-16 bit, 2-8 bit, 1-WDT	20	Yes	25mA source/sink per I/O, PSP, 2 CCP	40P, 40JW, 44L, 44PQ, 44PT
PIC16CR65	7168	–	4096x14	–	192	–	–	BOD	33	USART/I2C/SPI	2	1-16 bit, 2-8 bit, 1-WDT	20	Yes	25mA source/sink per I/O, PSP, 2 CCP	40P, 40JW, 44L, 44PQ, 44PT
PIC16C66	14336	8192x14	–	–	368	–	–	BOD	22	USART/I2C/SPI	2	1-16 bit, 2-8 bit, 1-WDT	20	Yes	25mA source/sink per I/O, 2 CCP	28SP, 28SO, 28JW
PIC16C67	14336	8192x14	–	–	368	–	–	BOD	33	USART/I2C/SPI	2	1-16 bit, 2-8 bit, 1-WDT	20	Yes	25mA source/sink per I/O, PSP, 2 CCP	40P, 40JW, 44L, 44PQ, 44PT
PIC16C620	896	512x14	–	–	80	–	2	BOD, Prog. VREF	13	–	–	1-8 bit, 1-WDT	20	Yes	25mA source/sink per I/O	18SP, 18SO, 20SS, 18JW
PIC16C620A	896	512x14	–	–	96	–	2	BOD, Prog. VREF	13	–	–	1-8 bit, 1-WDT	20	Yes	25mA source/sink per I/O	18SP, 18SO, 20SS, 18JW
PIC16CR620A	896	–	512x14	–	96	–	2	BOD, Prog. VREF	13	–	–	1-8 bit, 1-WDT	20	–	25mA source/sink per I/O	18SP, 18SO, 20SS
PIC16C621	1792	1024x14	–	–	80	–	2	BOD, Prog. VREF	13	–	–	1-8 bit, 1-WDT	20	Yes	25mA source/sink per I/O	18SP, 18SO, 20SS, 18JW
PIC16C621A	1792	1024x14	–	–	96	–	2	BOD, Prog. VREF	13	–	–	1-8 bit, 1-WDT	20	Yes	25mA source/sink per I/O	18SP, 18SO, 20SS, 18JW
PIC16C622	3584	2048x14	–	–	128	–	2	BOD, Prog. VREF	13	–	–	1-8 bit, 1-WDT	20	Yes	25mA source/sink per I/O	18SP, 18SO, 20SS, 18JW
PIC16C622A	3584	2048x14	–	–	128	–	2	BOD, Prog. VREF	13	–	–	1-8 bit, 1-WDT	20	Yes	25mA source/sink per I/O	18SP, 18SO, 20SS, 18JW
PIC16CE623	896	512x14	–	128	96	–	2	BOD, Prog. VREF	13	–	–	1-8 bit, 1-WDT	20	Yes	25mA source/sink per I/O	18SP, 18SO, 20SS, 18JW
PIC16CE624	1792	1024x14	–	128	96	–	2	BOD, Prog. VREF	13	–	–	1-8 bit, 1-WDT	20	Yes	25mA source/sink per I/O	18SP, 18SO, 20SS, 18JW
PIC16CE625	3584	2048x14	–	128	128	–	2	BOD, Prog. VREF	13	–	–	1-8 bit, 1-WDT	20	Yes	25mA source/sink per I/O	18SP, 18SO, 20SS, 18JW
PIC16F627*	1792 (Flash)	1024x14 (Flash)	–	128	224	–	2	BOD, Prog. VREF	16	USART/SCI	1	1-16 bit, 2-8 bit, 1-WDT	20	Yes	25mA source/sink per I/O 4MHz Internal Clock Oscillator, CCP	18SP, 18SO, 20SS
PIC16F628*	3584 (Flash)	2048x14 (Flash)	–	128	224	–	2	BOD, Prog. VREF	16	USART/SCI	1	1-16 bit, 2-8 bit, 1-WDT	20	Yes	25mA source/sink per I/O 4MHz Internal Clock Oscillator, CCP	18SP, 18SO, 20SS
PIC16C642	7168	4096x14	–	–	176	–	2	BOD, Prog. VREF	22	–	–	1-8 bit, 1-WDT	20	Yes	25mA source/sink per I/O	28SP, 28SO, 28JW
PIC16C662	7168	4096x14	–	–	176	–	2	BOD, Prog. VREF	33	–	–	1-8 bit, 1-WDT	20	Yes	25mA source/sink per I/O	40P, 40JW, 44L, 44PQ, 44PT
PIC16C710	896	512x14	–	–	36	4	–	BOD	13	–	–	1-8 bit, 1-WDT	20	Yes	25mA source/sink per I/O	18SP, 18SO, 20SS, 18JW
PIC16C71	1792	1024x14	–	–	36	4	–	–	13	–	–	1-8 bit, 1-WDT	20	Yes	25mA source/sink per I/O	18SP, 18SO, 20SS
PIC16C711	1792	1024x14	–	–	68	4	–	BOD	13	–	–	1-8 bit, 1-WDT	20	Yes	25mA source/sink per I/O	18SP, 18SO, 20SS, 18JW
PIC16C712*	1792	1024x14	–	–	128	4	–	BOD	13	–	1	1-16 bit, 2-8 bit, 1-WDT	20	Yes	CCP	18SP, 18SO, 20SS, 18JW
PIC16C715	3584	2048x14	–	–	128	4	–	BOD	13	–	–	1-8 bit, 1-WDT	20	Yes	25mA source/sink per I/O	18SP, 18SO, 20SS, 18JW
PIC16C716*	3584	2048x14	–	–	128	4	–	BOD	13	–	1	1-16 bit, 2-8 bit, 1-WDT	20	Yes	CCP	18SP, 18SO, 20SS, 18JW
PIC16C717*	3584	2048x14	–	–	256	6 (10-Bit	–	PBOD, LVD, Vref	16	I2C/SPI	1	1-16 bit, 2-8 bit, 1-WDT	20	Yes	4MHz internal oscillator, ECCP, MSSP	18P, 18SO, 18JW, 20SS
PIC16C72	3584	2048x14	–	–	128	5	–	BOD	22	I2C/SPI	1	1-16 bit, 2-8 bit, 1-WDT	20	Yes	25mA source/sink per I/O, CCP	28SP, 28SO, 28SS, 28JW

Tableau 1.3 – Caractéristiques résumées des circuits de la famille PIC 16CXXX (suite).

Référence	Mémoire de programme			Taille RAM		Numérique				Autres particularités		
	Octets	Taille OTP/Flash	Taille ROM			Ports d'E/S	E/S série	PWM 10 bits	Timers			
PIC16CXXX – 4-12 Interrupts, 200 ns Instruction Execution, 35 Instructions, 4/5 Oscillator Selections. Upwardly Compatible with PIC16C5X/PIC12CXXX												
PIC16C72A PIC16CR72	3584 3584	2048x14 –	– 2048x14	128 128	5 5	– –	BOD BOD	22 22	I2C/SPI I2C/SPI	1 1	1-16 bit, 2-8 bit, 1-WDT 1-16 bit, 2-8 bit, 1-WDT	20 20	Yes –	25mA source/sink per I/O, CCP 25mA source/sink per I/O, CCP		28SP, 28SO, 28SS, 28JW 28SP, 28SO, 28JW
PIC16C73A	7168	4096x14	–	192	5	–	BOD	22	USART/I2C/SPI	2	1-16 bit, 2-8 bit, 1-WDT	20	Yes	25mA source/sink per I/O, 2 CCP		28SP, 28SO, 28SS, 28JW
PIC16C73B	7168	4096x14	–	192	5	–	BOD	22	USART/I2C/SPI	2	1-16 bit, 2-8 bit, 1-WDT	20	Yes	25mA source/sink per I/O, 2 CCP		28SP, 28SO, 28SS, 28JW
PIC16C73C* PIC16CR73C*	7168 7168	4096x14 –	– 4096x14	192 192	5 5	– –	BOD BOD	22 22	USART/I2C/SPI USART/I2C/SPI	2 2	1-16 bit, 2-8 bit, 1-WDT 1-16 bit, 2-8 bit, 1-WDT	20 20	Yes Yes	25mA source/sink per I/O, PSP, 2 CCP 25mA source/sink per I/O, PSP, 2 CCP		28SP, 28SO, 28SS, 28JW 28SP, 28SO, 28JW
PIC16C74A	7168	4096x14	–	192	8	–	BOD	33	USART/I2C/SPI	2	1-16 bit, 2-8 bit, 1-WDT	20	Yes	25mA source/sink per I/O, PSP, 2 CCP		40P, 40JW, 44L, 44PQ, 44PT
PIC16C74B	7168	4096x14	–	192	8	–	BOD	33	USART/I2C/SPI	2	1-16 bit, 2-8 bit, 1-WDT	20	Yes	25mA source/sink per I/O, PSP, 2 CCP		40P, 40JW, 44L, 44PQ, 44PT
PIC16C74C* PIC16CR74C*	7168 7168	4096x14 –	– 4096x14	192 192	8 8	– –	BOD BOD	33 33	USART/I2C/SPI USART/I2C/SPI	2 2	1-16 bit, 2-8 bit, 1-WDT 1-16 bit, 2-8 bit, 1-WDT	20 20	Yes Yes	25mA source/sink per I/O, 2CCP, PSP 25mA source/sink per I/O, 2CCP, PSP		40P, 40JW, 44L, 44PQ, 44PT 40P, 44L, 44PQ, 44PT
PIC16C76	14336	8192x14	–	368	5	–	BOD	22	USART/I2C/SPI	2	1-16 bit, 2-8 bit, 1-WDT	20	Yes	25mA source/sink per I/O, 2CCP		28SP, 28SO, 28JW
PIC16C76A* PIC16CR76A*	14336 14336	8192x14 –	– 8192x14	368 368	5 5	– –	BOD BOD	22 22	USART/I2C/SPI USART/I2C/SPI	2 2	1-16 bit, 2-8 bit, 1-WDT 1-16 bit, 2-8 bit, 1-WDT	20 20	Yes Yes	25mA source/sink per I/O, 2CCP 25mA source/sink per I/O, 2CCP		28SP, 28SO, 28JW 28SP, 28SO, 28JW
PIC16C77	14336	8192x14	–	368	8	–	BOD	33	USART/I2C/SPI	2	1-16 bit, 2-8 bit, 1-WDT	20	Yes	25mA source/sink per I/O, PSP, 2CCP		40P, 40JW, 44L, 44PQ, 44PT
PIC16C77A* PIC16CR77A*	14336 14336	8192x14 –	– 8192x14	368 368	8 8	– –	BOD BOD	33 33	USART/I2C/SPI USART/I2C/SPI	2 2	1-16 bit, 2-8 bit, 3-WDT 1-16 bit, 2-8 bit, 3-WDT	20 20	Yes Yes	25mA source/sink per I/O, PSP, 2CCP 25mA source/sink per I/O, PSP, 2CCP		40P, 40JW, 44L, 44PQ, 44PT 40P, 40JW, 44L, 44PQ, 44PT
PIC16C745*	3584	2048x14	–	256	6 (12-bit	–	PBOD, LVD, Vref	16	I2C/SPI	1	1-16 bit, 2-8 bit, 1-WDT	20	Yes	4MHz internal oscillator, ECCP, MSSP		20P, 20SO, 20JW, 20SS
PIC16C765*	7168	4096x14	–	256	6 (12-bit	–	PBOD, LVD, Vref	16	I2C/SPI	1	1-16 bit, 2-8 bit, 1-WDT	20	Yes	4MHz internal oscillator, ECCP, MSSP		20P, 20SO, 20JW, 20SS
PIC16C770*	7168	4096x14	–	256	6 (12-bit)	–	PBOD & VREF & PLVD	22	USART/I2C/SPI	2	1-16 bit, 2-8 bit, 1-WDT	20	Yes	25mA source/sink per I/O, 2PSP, 2CCP		28SP, 28SO, 28SS, 28JW
PIC16C774*	7168	4096x14	–	256	10 (12-bit)	–	PBOD & Vref & PLVD	33	USART/I2C/SPI	2	1-16 bit, 2-8 bit, 1-WDT	20	Yes	25mA source/sink per I/O, PSP, 2CCP, MSSP, 9 bit		40P, 40JW, 44L, 44PQ, 44PT
PIC16C745*	14336	8192x14	–	256	5	–	BOD	22	USART	2	1-16 bit, 2-8 bit, 1-WDT	24	Yes	25mA source/sink per I/O, 2CCP, USB 1.1, 64 Bytes Dual Port RAM		28SP, 28SO, 28SS, 28JW
PIC16C765*	14336	8192x14	–	256	8	–	BOD	33	USART	2	1-16 bit, 2-8 bit, 1-WDT	24	Yes	25mA source/sink per I/O, 2CCP, USB 1.1, 64 Bytes Dual Port RAM, PSP		40P, 44L, 44PT, 44PQ, 64JW
PIC16F83 PIC16CR83	896 (Flash) 896	512x14 (Flash)	512x14	64 36	–	–	–	13 13	–	–	–	20 20	Yes	20mA source and 25mA sink per I/O 20mA source and 25mA sink per I/O		18P, 18SO 18P, 18SO
PIC16F84 PIC16CR84	1792 (Flash) 1792	1024x14 (Flash)	1024x14	68 68	–	–	–	13 13	–	–	–	20 20	Yes	20mA source and 25mA sink per I/O 20mA source and 25mA sink per I/O		18P, 18SO 18P, 18SO
PIC16F84A*	1792 (Flash)	1024x14 (Flash)	–	64	68	–	–	13	–	–	–	20	Yes	25mA source/sink per I/O		18P, 18SO, 20SS
PIC16F872*	3584 (Flash)	2048x14 (Flash)	–	64	128	5 pt-inte	–	BOD	22	PCI/SPI	1	1-16 bit, 2-8 bit, 1-WDT	20	Yes	CCP, MSSP		28P, 28SO, 28SS
PIC16F873*	7168 (Flash)	4096x14 (Flash)	–	128	192	5 (10-bit)	–	BOD	22	USART/I2C/SPI	2	1-16 bit, 2-8 bit, 3-WDT	20	Yes	25mA source/sink per I/O, CCP, MSSP, 9 bit		28SP, 28SO
PIC16F874*	7168 (Flash)	4096x14 (Flash)	–	128	192	8 (10-bit)	–	BOD	33	USART/I2C/SPI	2	1-16 bit, 2-8 bit, 1-WDT	20	Yes	25mA source/sink per I/O, PSP, CCP, MSSP, 9 bit		40P, 44L, 44PQ, 44PT
PIC16F876*	14336 (Flash)	8192x14 (Flash)	–	368	368	5 (10-bit)	–	BOD	22	USART/I2C/SPI	2	1-16 bit, 2-8 bit, 1-WDT	20	Yes	25mA source/sink per I/O, CCP, MSSP, 9 bit		28SP, 28SO
PIC16F877*	14336 (Flash)	8192x14 (Flash)	–	256	368	8 (10-bit)	–	BOD	33	USART/I2C/SPI	2	1-16 bit, 2-8 bit, 1-WDT	20	Yes	25mA source/sink per I/O, CCP, MSSP, 9 bit		40P, 44L, 44PQ, 44PT
PIC16C923	7168	4096x14	–	176	–	–	–	52	I2C/SPI	1	1-16 bit, 2-8 bit, 1-WDT	8	Yes	25mA source/sink per I/O, CCP, LCD module, static, 1/2, 1/3, 1/4 multiplex		64SP, 64PT
PIC16C924	7168	4096x14	–	176	5	–	–	52	I2C/SPI	1	1-16 bit, 2-8 bit, 1-WDT	8	Yes	25mA source/sink per I/O, CCP, LCD module, static, 1/2, 1/3, 1/4 multiplex		64SP, 68CL, 68L, 64PT

© Dunod. La photocopie non autorisée est un délit.

1.4 Dans la jungle des références

Avant d'aborder la présentation des différentes ressources des familles qui nous intéressent, attardons-nous quelques instants sur la méthode de construction des références des circuits ce qui, comme nous allons le voir, nous renseigne déjà sur un certain nombre de paramètres importants.

Une référence de microcontrôleur Microchip est toujours de la forme NN LLL XXX où :

- NN désigne la famille à laquelle appartient le circuit et peut donc valoir, à l'heure actuelle : 12, 16, 17 ou 18 ;
- LLL est un ensemble de une, deux ou trois lettres qui indique tout à la fois le type de mémoire de programme contenue dans le circuit et si l'on est présence d'un circuit à plage de tension d'alimentation normale ou étendue ;
- XXX est un ensemble de deux ou trois chiffres constituant la référence du circuit dans la famille et permettant de savoir de quelles ressources internes il dispose.

Un suffixe peut également suivre la référence, encore qu'en ce domaine un certain désordre semble régner chez Microchip puisque ces suffixes ont évolué au cours du temps. Comme nous le verrons lorsque nous parlerons de l'horloge, ils précisent quel type d'horloge peut être utilisé avec le circuit et surtout quelle est la fréquence maximale de cette dernière.

1.5 Les différentes mémoires de programme

La mémoire de programme est toujours le point noir des concepteurs de systèmes à base de microcontrôleurs car, si la mémoire programmable par masque est celle que l'on adopte toujours pour les productions en grandes séries, elle ne convient pas aux séries moins importantes et encore moins aux phases de développement.

Pour les petites séries, la mémoire OTPROM ou mémoire programmable une seule fois (*one time programmable ROM*) est idéale car elle n'impose pas de quota minimum de fabrication. En outre, sa programmation est généralement très simple et ne nécessite pas toujours d'investir dans un programmateur coûteux.

Pour les phases de développement, et si l'on ne dispose pas d'un émulateur, des versions avec de la mémoire effaçable, que ce soit électriquement [illegible] mémoire Flash ou aux ultraviolets, est évidemment une solution idéale.

Conscient de ces problèmes, Microchip propose évidemment ses circuits avec différents types de mémoires de programme. La distinction entre ces différentes versions de mémoires a lieu au moyen de la ou des lettres « centrales » de la référence comme indiqué ci-dessous :

- La lettre C correspond à de la mémoire de type EPROM qui peut donc être programmée et effacée électriquement, pour peu que le boîtier dont est muni le circuit soit pourvu d'une fenêtre en quartz afin de l'exposer aux ultraviolets. Dans le cas contraire (boîtier opaque en plastique ou céramique), ces circuits sont à considérer comme étant munis de mémoire de type OTPROM puisqu'il n'est possible de réaliser qu'une seule programmation, l'opacité du boîtier rendant tout effacement impossible.

- Le couple CR correspond à de la mémoire de programme de type ROM programmable par masque. Ces circuits ne peuvent donc être programmés que lors de leur phase de fabrication, ce qui les réserve à de la production en grande série.

- La lettre F enfin correspond à de la mémoire de type flash, c'est-à-dire à de la mémoire EPROM programmable et effaçable électriquement. Ces circuits peuvent donc être présentés en boîtier opaque (plastique ou céramique) tout en offrant des possibilités d'effacement. Ils supportent également la programmation en circuit sous forme série, ce qui permet de modifier le contenu de leur mémoire de programme sans devoir les enlever de l'application où ils sont installés.

Ces règles sont en principe respectées, mais vous rencontrerez peut-être encore des « vieux » 16C84 qui, malgré leur lettre C et un boîtier plastique opaque, sont effaçables électriquement. Ils sont aujourd'hui remplacés par les 16F84 avec lesquels ils sont interchangeables et dont l'appellation est plus conforme à ce que nous venons de voir.

Notez aussi que tous les microcontrôleurs des familles 12Cxxx et 16Cxxx n'existent pas dans toutes les versions de mémoires ; le choix proposé est cependant relativement vaste et propre à satisfaire la majorité des situations.

Quel que soit leur type, toutes les mémoires de programme des microcontrôleurs PIC disposent d'un bit de protection qui interdit leur relecture à partir d'un programmateur depuis l'extérieur du circuit, afin d'éviter la copie frauduleuse du logiciel contenu dans le circuit. Ce bit est positionné ou non par vos soins lors de la programmation du circuit. Il ne fait pas partie d'un registre interne et ne peut donc pas être manipulé par programme. De plus, sur les circuits à mémoire de programme effaçable, il ne peut être effacé qu'en même temps que l'effacement global de toute la mémoire. La confidentialité de votre programme est ainsi préservée d'un éventuel concurrent indélicat.

© Dunod. La photocopie non autorisée est un délit.

1.6 Tensions d'alimentation

Les microcontrôleurs PIC s'alimentent bien évidemment sous une tension unique (nous ne sommes pas revenus plus d'une décennie en arrière !) et le pluriel qui orne ce titre n'est là que pour que vous sachiez qu'il existe différentes versions de circuits avec des plages de tension d'alimentation différentes. Ici encore, c'est une lettre centrale de la référence qui permet de savoir dans quelle situation on se trouve :

- Les circuits ne disposant d'aucune lettre centrale particulière, autre que celle de désignation du type de mémoire vue précédemment, fonctionnent uniquement dans la plage de tension qui s'étend de 4,5 à 6,0 V.

- Les circuits affectés de la lettre L précédant la lettre type de mémoire fonctionnent dans une plage de tension étendue qui dépend du type de mémoire de programme dont est muni le circuit. Le *tableau 1.4* résume tout cela mieux qu'un long discours.

Tableau 1.4 – Les différentes plages de tensions d'alimentation et leurs lettres codes en fonction du type de mémoire de programme.

		EPROM		ROM		Flash	
Gamme de tension	Standard	C	4,5 à 6 V	CR	4,5 à 6 V	F	4,5 à 6 V
	Étendue	LC	2,5 à 6 V	LCR	2,5 à 6 V	LF	2 à 6 V

Dans tous les cas, les entrées/sorties des microcontrôleurs sont compatibles TTL lorsque ces derniers sont alimentés par une tension comprise dans la plage d'alimentation normalisée des circuits TTL, soit de 4,75 à 5,25 V.

Enfin, notez que la mémoire vive ou RAM contenue dans les microcontrôleurs PIC est capable de conserver son contenu lorsque le circuit est placé en mode sommeil jusqu'à une tension d'alimentation aussi basse que 1,5 V.

Dernière précision, mais qui va quasiment de soi en présence de circuits CMOS : le positif de la tension d'alimentation est appliqué à la patte V_{DD} du boîtier et la masse à la patte V_{SS}, ces deux pattes étant découplées au plus près du circuit au moyen d'un condensateur de 22 à 100 nF de type céramique multicouches.

1.7 Appellations normalisées des diverses pattes disponibles

Les pattes dont on dispose sur le boîtier d'un microcontrôleur PIC varient évidemment en fonction des ressources internes et de la taille de ce boîtier. Fort heureusement, chez Microchip la notion de famille de microcontrôleurs n'est pas un vain mot et tous les circuits utilisent les mêmes désignations de pattes à ressources internes identiques.

La patte PSP2, si elle est présente sur un boîtier, sera ainsi toujours la patte 2 du port parallèle esclave (PSP), quel que soit le type de circuit retenu. Cela facilite bien évidemment la prise de contact avec les différents circuits de la famille et évite d'avoir à se livrer à de fastidieuses conversions de noms.

Afin que vous ayez déjà une idée des ressources que vous allez rencontrer et des noms de pattes qui leur sont attachés, vous trouverez dans le *tableau 1.5* la liste complète de ces dernières et de leurs appellations, ainsi qu'un résumé de leurs fonctions et de leurs types électriques.

Les noms présents dans ce tableau sont bien évidemment ceux utilisés par Microchip dans ses documentations techniques afin que vous puissiez vous y repérer aisément. Ce sont aussi ceux que vous retrouverez tout au long de cet ouvrage lors de la description des ressources correspondantes.

Cela étant précisé, nous pouvons maintenant faire connaissance un peu plus en profondeur avec les microcontrôleurs PIC et nous allons commencer pour cela par les deux fonctions de base indispensables que sont les circuits d'horloge et d'initialisation (*reset*) car, même pour des fonctions aussi basiques, les PIC se distinguent !

© Dunod. La photocopie non autorisée est un délit.

Tableau 1.5 – Appellations, fonctions, types électrique et logique de toutes les pattes pouvant être rencontrées sur les différentes versions de PIC.

Appellation	Type logique	Type électrique	Fonction
AN0 à AN15	E	Analogique	Entrées analogiques
AV_{DD}	Alimentation	Alimentation	Alimentation étages analogiques
AV_{SS}	Masse	Masse	Masse étages analogiques
C1	E	Analogique	Convertisseur à pompe de charge de l'interface LCD
C2	E	Analogique	Convertisseur à pompe de charge de l'interface LCD
CCP1	E/S	Trigger de Schmitt	E/S module CCP1, PWM1
CCP2	E/S	Trigger de Schmitt	E/S module CCP2, PWM2
CK	E/S	Trigger de Schmitt	Horloge synchrone SCI
CLKIN/OSC1	E	Trigger de Schmitt ou CMOS	Oscillateur d'horloge
CLKOUT/OSC2	S	-	Oscillateur d'horloge
CMPA	S	-	Comparateur analogique A
CMPB	S	-	Comparateur analogique B
COM0	S	Driver LCD	Électrode commune 0 afficheur LCD
COM1	S	Driver LCD	Électrode commune 1 afficheur LCD
COM2	S	Driver LCD	Électrode commune 2 afficheur LCD
COM3	S	Driver LCD	Électrode commune 3 afficheur LCD
DT	E/S	Trigger de Schmitt	Données synchrones SCI
GP0	E/S	TTL/Trigger de Schmitt	Ligne 0 port GPIO
GP1	E/S	TTL/Trigger de Schmitt	Ligne 1 port GPIO
GP2	E/S	Trigger de Schmitt	Ligne 2 port GPIO
GP3	E	TTL	Ligne 3 port GPIO
GP4	E/S	TTL	Ligne 4 port GPIO

Appellation	Type logique	Type électrique	Fonction
GP5	E/S	TTL	Ligne 5 port GPIO
INT	E	Trigger de Schmitt	Interruption externe
$\overline{MCLR}$ / V_{PP}	E/Alimentation	Trigger de Schmitt	Entrée de *Reset* et de tension V_{PP} en mode programmation
PSP0 à PSP7	E/S	TTL	Lignes 0 à 7 PSP
RA0 à RA3	E/S	TTL	Lignes 0 à 3 port A
RA4	E/S	Trigger de Schmitt	Ligne 4 port A
RA5	E/S	TTL	Ligne 5 port A
RB0 à RB7	E/S	TTL	Lignes 0 à 7 port B
RC0 à RC7	E/S	Trigger de Schmitt	Lignes 0 à 7 port C
RD0 à RD7	E/S	Trigger de Schmitt	Lignes 0 à 7 port D
RE0 à RE7	E/S	Trigger de Schmitt	Lignes 0 à 7 port E
RF0 à RF7	E/S	Trigger de Schmitt	Lignes 0 à 7 port F
RG0 à RG7	E/S	Trigger de Schmitt	Lignes 0 à 7 port G ou sorties segments port LCD
$\overline{CS}$	E	TTL	$\overline{CS}$ en mode PSP
$\overline{RD}$	E	TTL	Entrée lecture en mode PSP
$\overline{WR}$	E	TTL	Entrée écriture en mode PSP
REFA	S	CMOS	Sortie référence A
REFB	S	CMOS	Sortie référence B
RX	E	Trigger de Schmitt	Réception asynchrone SCI
TX	S	-	Émission asynchrone SCI
SCL, SCLA, SCLB	E/S	Trigger de Schmitt	Horloge pour bus I2C
SDA, SDAA, SDAB	E/S	Trigger de Schmitt	Données pour bus I2C
SCK	E/S	Trigger de Schmitt	Horloge en mode SPI

Appellation	Type logique	Type électrique	Fonction
SDI	E	Trigger de Schmitt	Entrée de données en mode SPI
SDO	S	-	Sortie de données en mode SPI
$\overline{SS}$	E	Trigger de Schmitt	Sélection maître/esclave en mode SPI
SEG00 à SEG31	S	LCD	Sorties segments port LCD
T0CKI	E	Trigger de Schmitt	Horloge externe *timer 0*
T1CKI	E	Trigger de Schmitt	Horloge externe *timer 1*
T1OSO	S	CMOS	Oscillateur *timer 1*
T1OSI	E	CMOS	Oscillateur *timer 1*
V_{LCD1} à V_{LCD3}	Alimentation	-	Alimentation afficheur LCD
VLCDADJ	E	Analogique	Réglage alimentation afficheur LCD
V_{REF}	E/S	Analogique	Entrée tension de référence convertisseur A/D, sortie tension de référence du générateur interne
V_{REF+}	E	Analogique	Entrée tension de référence haute convertisseur A/D
V_{REF-}	E	Analogique	Entrée tension de référence basse convertisseur A/D
V_{SS}	Alimentation	-	Masse du circuit
V_{DD}	Alimentation	-	Alimentation positive du circuit

CHAPITRE 2

HORLOGE ET *RESET*

Même si ce ne sont pas là les deux fonctions les plus nobles d'un microcontrôleur, aucun circuit ne peut s'en passer. Nous avons donc choisi de commencer cette visite guidée au monde des PIC par la présentation des deux piliers de l'unité centrale que sont la circuiterie de génération d'horloge et celle d'initialisation, plus communément appelée de *reset*, même si cela fait frémir ces messieurs de l'Académie.

2.1 Les horloges des PIC

2.1.1 Les différents types d'horloges

Toutes versions de circuits confondues, ce ne sont pas moins de huit types d'horloges différents que nous propose, ou plutôt que prétend nous proposer, Microchip. Elles sont repérées par une appellation normalisée que nous avons bien évidemment conservée afin que vous ne soyez pas dépaysé si vous consultez les fiches techniques des circuits. On dispose donc des types suivants :

- LP pour l'horloge à quartz basse vitesse ; LP est l'abréviation de *low power* car cette horloge est celle qui fait consommer le moins de courant au circuit en raison de sa lenteur ;
- XT pour l'horloge à quartz ou à résonateur céramique standard ;
- HS pour l'horloge à quartz ou à résonateur céramique haute vitesse (HS pour *high speed*) ;
- RC pour l'horloge pilotée par un ensemble résistance-capacité ;
- EXTRC pour l'horloge pilotée par un ensemble résistance-capacité externe ;
- EXTRC pour l'horloge pilotée par un ensemble résistance-capacité externe avec sortie d'horloge sur la patte CLKOUT ;

© Dunod. La photocopie non autorisée est un délit.

- INTRC pour l'horloge pilotée par un oscillateur à résistance-capacité interne ;
- INTRC pour l'horloge pilotée par un oscillateur à résistance-capacité interne avec sortie d'horloge sur la patte CLKOUT.

Comme ces courtes descriptions le laissent entendre, un certain nombre d'horloges RC ne sont en fait que des variantes mineures les unes des autres, d'où nos réserves précédentes quant aux soi-disant huit types d'horloges différents !

Ce type d'horloge est librement programmable au moyen de deux ou trois bits de configuration du circuit, appelés FOSC0, FOSC1 et FOSC2 comme indiqué dans le *tableau 2.1*.

Tableau 2.1 – Programmation des bits FOSC0 à FOSC2 en fonction du type d'oscillateur désiré.

FOSC2 :FOSC1 :FOSC0	Mode	Commentaire
111	EXTRC avec CLKOUT	Solution économique Fréquence peu précise Sortie sur CLKOUT Mode par défaut
110	EXTRC	Idem que ci-dessus mais CLKOUT invalidée et disponible pour E/S à usage général
101	INTRC avec CLKOUT	Solution très économique Horloge à 4 MHz avec ajustement possible Sortie sur CLKOUT
100	INTRC	Solution très économique Horloge à 4 MHz avec ajustement possible CLKOUT invalidée et disponible pour E/S à usage général
011	-	Réservé
010	HS	Oscillateur à quartz ou résonateur céramique haute fréquence Consommation élevée
001	XT	Oscillateur à quartz ou à résonateur céramique moyenne fréquence Consommation moyenne
000	LP	Oscillateur à quartz très basse fréquence Consommation très faible

Toutefois, sur les versions de circuits offrant le moins de choix, le bit FOSC2 est absent et on ne dispose plus alors que des quatre premiers types d'horloges présentés ci-dessus comme le confirme le *tableau 2.2*.

© Dunod. La photocopie non autorisée est un délit.

Tableau 2.2 – Programmation des bits FOSC0 et FOSC1 en fonction du type d'oscillateur désiré.

FOSC1 :FOSC0	Mode	Commentaire
11	RC	Solution économique Fréquence peu précise Mode par défaut
10	HS	Oscillateur à quartz ou résonateur céramique haute fréquence Consommation élevée
01	XT	Oscillateur à quartz ou à résonateur céramique moyenne fréquence Consommation moyenne
00	LP	Oscillateur à quartz très basse fréquence Consommation très faible

Ces tableaux appellent un commentaire sur les différents oscillateurs à quartz que sont le LP, le XT et le HS. En fait, ils ne diffèrent que par le gain de l'inverseur interne utilisé pour réaliser l'oscillateur d'horloge. Plus l'horloge doit être rapide, plus ce gain doit être élevé et, accessoirement, plus la consommation du circuit augmente. Pour ce faire, il importe de toujours choisir, grâce aux bits FOSC0 à FOSC2, la valeur de gain la plus faible possible compatible avec la fréquence du quartz choisie. La fiche technique de chaque circuit précise les limites de fonctionnement de chaque mode.

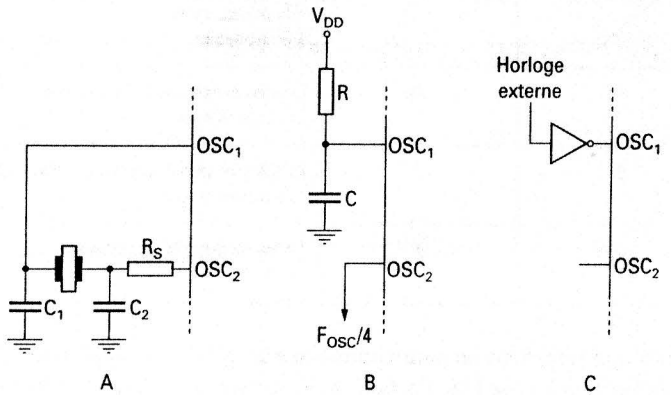
Notez que ces bits ne sont contenus dans aucun registre en tant que tel. Ils font partie des bits de configuration générale du circuit et, de ce fait, vous n'aurez que très rarement à les manipuler directement. En effet, tous les bons programmateurs de PIC assurent un « décodage » de ces derniers et, plutôt que de vous demander de définir l'état de FOSC0 à FOSC2, ils vous demandent si vous souhaitez une horloge de type LP, XT, etc. C'est tout de même plus parlant !

2.1.2 Les schémas d'horloges types

Une fois défini le type d'horloge désiré, encore faut-il connecter correctement les éléments externes aux pattes disponibles sur le boîtier, ce qui est à faire en respectant les indications que voici.

Deux pattes d'horloge sont présentes sur tous les boîtiers des diverses familles de PIC et s'utilisent de la même façon : ce sont OSC1 et OSC2/CLKOUT. Notez toutefois dès à présent que ces pattes peuvent être absentes sur les circuits 12Cxxx lorsque l'on choisit de les faire fonctionner en mode horloge interne, afin de disposer ainsi d'un maximum de pattes d'entrées/sorties sur ces boîtiers. Lorsqu'elles sont présentes, elles fonctionnent comme sur tous les autres microcontrôleurs.

Compte tenu des différents types d'horloges utilisables avec les microcontrôleurs PIC, il existe trois schémas principaux de connexion des éléments externes à l'oscillateur, schémas résumés en *figure 2.1*.



Osc. type	Fréquence	C_1	C_2
LP	32 kHz§ 100 kHz 200 kHz	15 pF 15 pF 0-15 pF	15 pF 15 pF 0-15 pF
XT	100 kHz 200 kHz 455 kHz 1 MHz 2 MHz 4 MHz	15-30 pF 15-30 pF 15-30 pF 15-30 pF 15 pF 15 pF	200-300 pF 100-200 pF 15-100 pF 15-30 pF 15 pF 15 pF
HS	4 MHz 8 MHz 20 MHz	15 pF 15 pF 15 pF	15 pF 15 pF 15 pF

§ pour V_{DD} > 4,5 V, $C_1 = C_2 = 30$ pF

Figure 2.1 – Les trois schémas d'horloge utilisables avec les PIC 12Cxxx et 16Cxxx.

Pour les versions à quartz ou à résonateur céramique, on utilise le schéma A de la *figure 2.1*. La valeur des condensateurs C_1 et C_2 varie selon que l'on est en présence d'un quartz ou d'un résonateur et également selon la fréquence de fonctionnement, conformément aux indications du tableau visible sur cette même figure. La résistance R_s n'est en principe pas nécessaire avec les versions XT mais

doit parfois être présente avec les versions HS. Seules les caractéristiques exactes du quartz permettent de savoir si elle est ou non nécessaire et quelle doit être sa valeur.

Le schéma de câblage de la version RC est un peu plus original, comme le montre le schéma B *figure 2.1*, et fait plus penser à une circuiterie de *reset* qu'à un oscillateur d'horloge. C'est pourtant là le mode de connexion à utiliser, qui présente en outre l'avantage de fournir sur la patte OSC2/CLKOUT une sortie à 1/4 de la fréquence d'oscillation, sous réserve que l'on ait programmé correctement les bits FOSC0 à FOSC2 de façon à sélectionner le mode RC, ou le mode EXTRC avec sortie d'horloge sur CLKOUT.

La stabilité de ce type d'horloge est évidemment limitée mais, pour qu'elle ne soit pas trop influencée par les éléments extérieurs, Microchip recommande une valeur de *R* comprise entre 3 kΩ et 100 kΩ et un condensateur d'une valeur supérieure à 20 pF afin de s'affranchir de l'influence des capacités parasites du boîtier et du câblage.

Le schéma C *figure 2.1* enfin montre le dernier mode d'utilisation de l'horloge des PIC, qui est le pilotage à partir d'un oscillateur externe. Cet oscillateur peut être de n'importe quel type, pour peu qu'il délivre des niveaux logiques compatibles avec la tension d'alimentation du microcontrôleur.

2.1.3 L'horloge RC interne et son registre de calibration

Contrairement à l'horloge à quartz ou à circuit RC externe, qui peut être utilisée avec tous les microcontrôleurs de la gamme, l'horloge RC interne n'est pas disponible sur tous les produits. Les différents tableaux de sélection publiés dans cet ouvrage mentionnent donc explicitement cette possibilité dans la colonne « particularités ».

Lorsque cette horloge existe, elle fonctionne à 4 MHz, ne nécessite évidemment aucun composant externe, mais offre en plus une ressource particulière sous la forme d'un registre interne de calibration qui permet de régler sa fréquence de fonctionnement exacte avec une bonne précision. En effet, vous savez sans doute que la diffusion de composants passifs sur les puces de circuits intégrés conduit à des tolérances assez larges sur les valeurs des éléments obtenus et, si l'on ne prenait pas de précaution, la fréquence réelle de cette horloge serait parfois assez éloignée de sa valeur théorique de 4 MHz.

Les PIC munis de cette horloge disposent donc d'un registre appelé OSCCAL qui permet d'ajuster cette fréquence. Lors de la fabrication des circuits, Microchip vous communique, d'une façon un peu particulière que nous allons décrire ci-dessous, la valeur à programmer dans ce registre afin d'amener la fréquence de l'horloge au plus près des 4 MHz théoriques.

© Dunod. La photocopie non autorisée est un délit.

Si l'exactitude de la fréquence d'horloge est importante pour votre application, il est donc indispensable de tenir compte de cette information afin de la programmer dans le registre OSCCAL. Pour cela deux procédures existent selon les circuits, mais elles conduisent toutes deux aux mêmes résultats.

Pour les PIC 12Cxxx, on trouve, à l'adresse la plus haute de la mémoire de programme, une instruction MOVLW XX, où XX représente justement la donnée à placer dans le registre OSCCAL. Lors d'un *reset*, cette valeur est copiée dans le registre W du PIC et le compteur de programme ou PC est ensuite initialisé afin de lancer l'exécution de votre programme. Vous pouvez alors, au moyen de la première instruction de ce programme, copier le contenu de W dans OSCCAL afin d'ajuster correctement l'horloge.

Pour les PIC 16Cxxx, on trouve à l'adresse la plus haute de la mémoire de programme une instruction RETLW XX, où XX est la donnée à placer dans le registre OSCCAL. Au début du programme utilisateur, il suffit alors de placer un CALL YY, où YY représente l'adresse la plus haute de la mémoire de programme, pour récupérer la valeur XX dans le registre W et pouvoir ensuite l'écrire dans le registre OSCCAL.

Cette façon de procéder nous conduit à attirer votre attention sur deux points :

- Il ne faut en aucun cas effacer la mémoire de programme de votre circuit (pour peu qu'elle soit effaçable) avant d'avoir pris connaissance, au moyen d'un programme adéquat, de la valeur à programmer dans OSCCAL. Cette valeur n'est en effet codée qu'au niveau du MOVLW ou du RETLW XX placé en haut de la mémoire de programme, et nulle part ailleurs. Elle est donc irrémédiablement détruite par l'effacement de cette mémoire et la seule méthode pour la retrouver ensuite, si vous ne l'avez pas notée, est de procéder par approximations successives, ce qui peut s'avérer assez long.
- Même si le bit de protection de la mémoire de programme en lecture (voir plus avant dans cet ouvrage) est programmé, la dernière adresse de cette mémoire reste lisible afin que vous puissiez toujours prendre connaissance de la valeur à programmer dans OSCCAL.

Cela étant précisé, voici néanmoins la signification des différents bits contenus dans le registre OSCCAL pour le cas où vous souhaiteriez intervenir sur ce dernier afin de modifier la fréquence d'horloge (ou pour retrouver la valeur non notée parce que vous n'avez pas suivi les conseils précédents...).

- ***Bit 0 et bit 1***

  Bits non utilisés et lus comme étant à 0. Ces bits ne servent à rien dans les versions actuelles des circuits, mais ils doivent être programmés à 0 lors de l'écriture dans le registre OSCCAL afin d'assurer une compatibilité ascendante avec les futures versions de PIC.

© Dunod. La photocopie non autorisée est un délit.

Accès E : Écriture L : Lecture	L/E	L/E	L/E	L/E	L/E	L/E	L	L
OSCCAL	CAL3	CAL2	CAL1	CAL0	CALFST	CALSLW	—	—
État à la mise sous tension	0	1	1	1	0	0	0	0

- ***Bit 2 ou bit CALSLW***

  S'il est mis à 0, ce bit est sans effet. S'il est mis à 1, il décale la plage de réglage de la fréquence de l'oscillateur d'horloge, accessible au moyen des bits 4 à 7, vers le bas.

- ***Bit 3 ou bit CALFST***

  S'il est mis à 0, ce bit est sans effet. S'il est mis à 1, il décale la plage de réglage de la fréquence de l'oscillateur d'horloge, accessible au moyen des bits 4 à 7, vers le haut. Si ce bit est mis à 1 en même temps que le bit 2, il est prioritaire sur ce dernier.

- ***Bits 4 à 7 ou bits CAL0 à CAL3***

  Ces bits servent à programmer le décalage de la fréquence d'horloge par rapport au centre de la fenêtre de calibration. C'est cette fenêtre qui peut, quant à elle, être décalée vers le haut ou vers le bas au moyen des bits 2 et 3. Suite à un *reset*, ces bits sont initialisés à la valeur décimale 7 (soit 0111 en binaire) afin de placer le décalage au centre de la fenêtre.

2.1.4 Comment récupérer le contenu de OSCCAL ?

Si, malgré tous nos avertissements, vous n'avez pas noté le contenu de OSCCAL avant d'effacer la mémoire de programme, la méthode la plus simple pour le déterminer est la suivante :

- configurez le circuit en mode horloge RC interne avec sortie sur CLKOUT comme nous l'avons vu ci-dessus ;
- placez un fréquencemètre sur la sortie CLKOUT ;
- écrivez un programme qui incrémente pas-à-pas les bits CAL0 à CAL3 (puis éventuellement les bits de décalage) du registre OSCCAL et qui soit capable de s'arrêter à la demande et d'indiquer le contenu atteint à cet instant par OSCCAL (par exemple au moyen de LED branchées sur un port parallèle) ;

– lancez l'exécution de ce programme et, lorsque le fréquencemètre indique 1 MHz (la sortie CLKOUT correspondant à la fréquence d'horloge interne divisée par quatre), le registre OSCCAL contient la bonne valeur qu'il ne vous reste plus alors qu'à lire sur l'organe de sortie prévu.

2.2 Le *reset*

2.2.1 Principes généraux

Même pour une fonction aussi banale que l'initialisation ou *reset*, les microcontrôleurs PIC arrivent à se faire remarquer. Ils peuvent en effet disposer de six sources de *reset* distinctes avec :

– un *reset* à la mise sous tension du circuit appelé POR pour *power on reset* ;

– un *reset* par action sur la patte $\overline{\text{MCLR}}$ alors que le circuit est en mode normal ;

– une *reset* par action sur la patte $\overline{\text{MCLR}}$ alors que le circuit est en mode sommeil ;

– un *reset* par débordement du *timer* chien de garde (WDT pour *watch dog timer*) alors que le circuit est en mode normal ;

– un *reset* par détection d'une chute de tension d'alimentation, même temporaire, appelé BOR pour *brown out reset* ;

– un *reset* suite à une erreur de parité mémoire appelé PER pour *parity error reset.*

Notez dès à présent que tous les microcontrôleurs de la famille ne disposent pas de toutes ces sources de *reset* puisque les deux dernières, c'est-à-dire le BOR et le PER, ne sont présentes que sur certains circuits comme cela est indiqué au moyen de ces mêmes appellations dans les tableaux de sélection présentés au chapitre 1.

Le comportement du circuit et l'état des registres affectés par un *reset* sont différents selon la situation qui se produit. Tout d'abord, il est possible de distinguer par logiciel l'origine du *reset*. Il suffit pour cela de lire les bits appropriés du registre d'état du microcontrôleur. Le *tableau 2.3* montre quels sont les bits concernés dans ce registre et leurs significations respectives.

Dans tous les cas de *reset*, sauf réveil du mode sommeil par le *timer* chien de garde ou l'arrivée d'une interruption, le compteur ordinal ou PC est mis à 000. La première instruction exécutable de votre programme doit donc se trouver à cette adresse. En cas de réveil d'un mode sommeil par le *timer* chien de garde, le PC est simplement augmenté de un pour passer à l'instruction qui suit l'instruction SLEEP. Tout cela est résumé de façon synthétique et visuelle dans le *tableau 2.4.*

© Dunod. La photocopie non autorisée est un délit.

Tableau 2.3 – État des bits des registres PCON et STATUS en fonction du type de *reset*.

POR	BOR(1)	TO	PD	Condition
0	X	1	1	Mise sous tension (POR)
0	X	0	X	Invalide
0	X	X	0	Invalide
1(2)	0	1	1	Baisse de tension d'alimentation (BOR)
1(2)	1(2)	0	1	*Reset* par le *timer* chien de garde (WDT)
1(2)	1(2)	0	0	Réveil par le *timer* chien de garde (WDT)
1(2)	1(2)	U	U	Action sur MCLR en mode normal
1(2)	1(2)	1	0	Action sur MCLR en mode sommeil

U - non modifié.
X - indéterminé.
(1) Tous les circuits ne disposent pas de la fonction BOR, ce bit peut donc être absent.
(2) Ces bits ne sont pas modifiés et, lorsqu'ils ont été initialisés suite à un *reset* de type POR ou BOR, ils sont lus comme étant à 1.

Tableau 2.4 – État des registres PCON, STATUS et du PC en fonction du type de *reset*.

Condition	PC	STATUS	PCON
Mise sous tension (POR)	000	0001 1XXX	U---10X
Action sur MCLR en mode normal	000	000U UUUU	U---UUU
Action sur MCLR en mode sommeil	000	0001 0UUU	U---UUU
Reset par le *timer* chien de garde (WDT)	000	0000 1UUU	U---UUU
Réveil par le *timer* chien de garde (WDT)	PC + 1	UUU0 0UUU	U---UUU
Baisse de tension d'alimentation (BOR)	000	0001 1UUU	U---UU0
Réveil du mode sommeil par une interruption	PC + 1 ou 004 (voir chapitre 5)	UUU1 0UUU	U---UUU

U - non modifié.
X - indéterminé.

Les contenus des autres registres de contrôle, d'état ou de données du microcontrôleur sont diversement affectés par les différentes sources de *reset* possibles. Nous indiquons, tout au long de l'ouvrage lors de la description des bits des registres internes, leur état suite à un *reset* de mise sous tension, mais vous trouvez dans le *tableau 2.5* ces mêmes indications, beaucoup plus complètes, pour les diverses situations de *reset* possibles.

Tableau 2.5 – État des bits des principaux registres internes suite à différents types de *reset*.

Registre	Mise sous tension	*Reset* MCLR pendant : – fonctionnement normal ; – mode *sleep*. Débordement chien de garde en fonctionnement normal	Réveil du mode *sleep* par : – interruption ; – chien de garde.
ADCAPL	0000 0000	0000 0000	uuuu uuuu
ADCAPH	0000 0000	0000 0000	uuuu uuuu
ADCON0	0000 00-0	0000 00-0	uuuu uu-u
ADCON1	----000	----000	----uuu
ADRES	xxxx xxxx	uuuu uuuu	uuuu uuuu
ADTMRL	0000 0000	0000 0000	uuuu uuuu
ADMRH	0000 0000	0000 0000	uuuu uuuu
CCP1CON	--00 0000	--00 0000	--uu uuuu
CCP2CON	0000 0000	0000 0000	uuuu uuuu
CCPR1L	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCPR1H	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCPR2L	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCPR2H	xxxx xxxx	uuuu uuuu	uuuu uuuu
CMCON	00-- 0000	00-- 0000	uu-- uuuu
EEADR	xxxx xxxx	uuuu uuuu	uuuu uuuu
EECON1	---0 x000	---0 q000	---0 uuuu
EECON2	-	-	-
EEDATA	xxxx xxxx	uuuu uuuu	uuuu uuuu
FSR	xxxx xxxx	uuuu uuuu	uuuu uuuu
GPIO	--xx xxxx	--uu uuuu	--uu uuuu
I2CADD	0000 0000	0000 0000	uuuu uuuu
I2CBUF	xxxx xxxx	uuuu uuuu	uuuu uuuu

© Dunod. La photocopie non autorisée est un délit.

Registre	Mise sous tension	*Reset* MCLR pendant : – fonctionnement normal ; – mode *sleep*. Débordement chien de garde en fonctionnement normal	Réveil du mode *sleep* par : – interruption ; – chien de garde.
I2CCON	0000 0000	0000 0000	uuuu uuuu
I2CSTAT	--00 0000	--00 0000	--uu uuuu
INDF	-	-	-
INTCON	0000 000x	0000 000u	uuuu uuuu(1)
LCDCON	00-0 0000	00-0 0000	uu-u uuuu
LCDD00 to LCDD15	xxxx xxxx	uuuu uuuu	uuuu uuuu
LCDPS	---- 0000	---- 0000	---- uuuu
LCDSE	1111 1111	1111 1111	uuuu uuuu
OPTION_REG	1111 1111	1111 1111	uuuu uuuu
OSCCAL	0111 00--	uuuu uu--	uuuu uu--
PCL	0000 0000	0000 0000	PC + 1 (2)
PCLATH	---0 0000	---0 0000	---u uuuu
PCON	-----0u	-----uu	-----uu
PIE1	0000 0000	0000 0000	uuuu uuuu
PIE2	-------0	-------0	-------u
PIR1	0000 0000	0000 0000	uuuu uuuu
PIR2	-------0	-------0	-------u
PORTA	--xx xxxx	--uu uuuu	--uu uuuu
PORTB	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTC	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTD	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTE	----xxx	----uuu	----uuu
PORTF	0000 0000	0000 0000	uuuu uuuu
PORTG	0000 0000	0000 0000	uuuu uuuu
PR2	1111 1111	1111 1111	1111 1111
PREFA	0000 0000	0000 0000	uuuu uuuu
PREFB	0000 0000	0000 0000	uuuu uuuu
RCSTA	0000 -00x	0000 -00x	uuuu -uuu
RCREG	0000 0000	0000 0000	uuuu uuuu
SLPCON	0011 1111	0011 1111	uuuu uuuu

Registre	Mise sous tension	*Reset* MCLR pendant : – fonctionnement normal ; – mode *sleep*. Débordement chien de garde en fonctionnement normal	Réveil du mode *sleep* par : – interruption ; – chien de garde.
SPBRG	0000 0000	0000 0000	uuuu uuuu
SSPBUF	xxxx xxxx	uuuu uuuu	uuuu uuuu
SSPCON	0000 0000	0000 0000	uuuu uuuu
SSPADD	0000 0000	0000 0000	uuuu uuuu
SSPSTAT	0000 0000	0000 0000	uuuu uuuu
STATUS	0001 1xxx	000q quuu(3)	uuuq quuu(3)
T1CON	--00 0000	--uu uuuu	--uu uuuu
T2CON	-000 0000	-000 0000	-uuu uuuu
TMR0	xxxx xxxx	uuuu uuuu	uuuu uuuu
TMR1L	xxxx xxxx	uuuu uuuu	uuuu uuuu
TMR1H	xxxx xxxx	uuuu uuuu	uuuu uuuu
TMR2	0000 0000	0000 0000	uuuu uuuu
TRIS	--11 1111	--11 1111	--uu uuuu
TRISA	--11 1111	--11 1111	--uu uuuu
TRISB	1111 1111	1111 1111	uuuu uuuu
TRISC	1111 1111	1111 1111	uuuu uuuu
TRISD	1111 1111	1111 1111	uuuu uuuu
TRISE	0000 -111	0000 -111	uuuu -uuu
TRISF	1111 1111	1111 1111	uuuu uuuu
TRISG	1111 1111	1111 1111	uuuu uuuu
TXREG	0000 0000	0000 0000	uuuu uuuu
TXSTA	0000 -010	0000 -010	uuuu -uuu
VRCON	000- 0000	000- 0000	uuu- uuuu
W	xxxx xxxx	uuuu uuuu	uuuu uuuu

u - bit non modifié.
x - état indéterminé.
- - bit inexistant, lu comme étant à 0.
q - l'état dépend de diverses autres conditions.
(1) Un ou plusieurs bits des registres INTCON et PIR1 peuvent être affectés.
(2) Lorsque le réveil est dû à une interruption et que le bit GIE est à 1, le PC est chargé avec 004 (voir chapitre 5).
(3) Voir *tableau 2.3*.

© Dunod. La photocopie non autorisée est un délit.

Même si ce tableau est assez indigeste, son examen est quasiment indispensable avant toute exploitation d'un registre suite à un *reset*, sauf bien sûr si votre programme décide de réinitialiser lui-même chaque registre avant toute utilisation.

2.2.2 Circuiterie de *reset* externe

Tous les circuits des familles 12Cxxx et 16Cxxx adoptent la même démarche, à savoir la présence d'une seule patte de *reset* baptisée $\overline{\text{MCLR}}$. Notez toutefois dès à présent que cette patte peut être absente sur les circuits 12Cxxx lorsque l'on choisit de les faire fonctionner en mode *reset* interne automatique à la mise sous tension, afin de disposer ainsi d'un maximum de pattes d'entrées/sorties sur ces boîtiers. Lorsqu'elle est présente, elle fonctionne comme sur tous les autres microcontrôleurs.

Cette patte de *reset* dispose de quelques particularités pour qui est habitué aux microcontrôleurs classiques. En effet, ces circuits intègrent en interne une circuiterie de *reset* automatique à la mise sous tension qui, si cette procédure s'avère suffisante (pas de besoin de *reset* externe manuel par exemple) et si la vitesse de croissance de la tension d'alimentation est assez élevée (typiquement supérieure à 0,05 $V \cdot ms^{-1}$), se suffit à elle-même. Le câblage du circuit de *reset* se résume donc à ce que vous pouvez voir *figure 2.2A* : la patte $\overline{\text{MCLR}}$ est tout simplement reliée à la tension d'alimentation positive V_{DD}, soit directement, soit au moyen d'une résistance optionnelle de quelques kilo-ohms.

Si cette vitesse de croissance de la tension d'alimentation n'est pas atteinte, si vous avez besoin d'une commande de *reset* externe ou bien encore si vous utilisez un quartz de fréquence relativement basse, dont le temps de mise en oscillation est donc important, la circuiterie de la *figure 2.2B* doit alors être mise en œuvre.

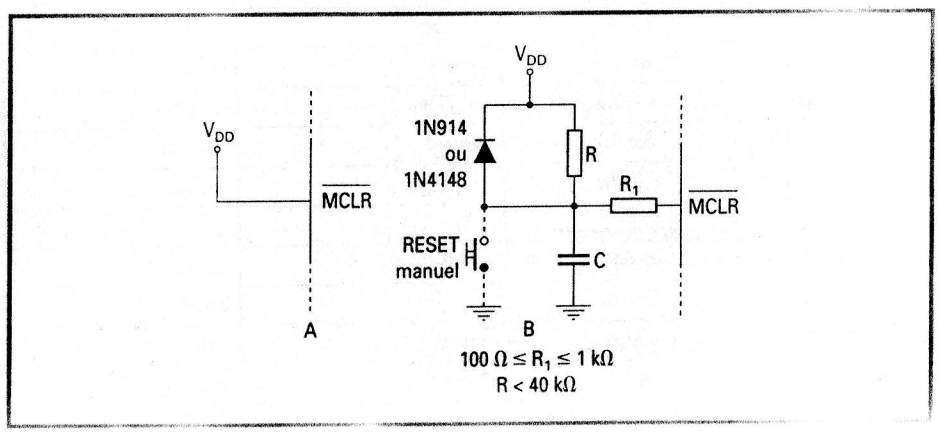


Figure 2.2 – Reset automatique à la mise sous tension (A) et reset manuel (B).

On retrouve ici un schéma connu des utilisateurs de microcontrôleurs classiques ! Notez la présence de la résistance R_1 dont la valeur varie de 100 Ω à 1 kΩ. Elle a pour but de protéger le circuit contre un courant de décharge de C dans la patte MCLR en cas de décharge électrostatique à proximité.

Enfin, si la tension d'alimentation est susceptible de varier dans des proportions telles que le bon fonctionnement du circuit peut être compromis, il est prudent de prévoir une circuiterie de *reset* capable de déclencher ce dernier, si justement l'alimentation descend en dessous d'un certain seuil. Pour les PIC disposant de la fonction BOR pour *brown out reset*, clairement identifiés en tant que tels dans les tableaux de sélection du chapitre 1, aucun composant externe supplémentaire n'est nécessaire, comme nous le verrons dans un instant.

Pour les circuits qui ne disposent pas d'une telle fonction, deux schémas sont proposés par Microchip et vous sont présentés *figure 2.3A* et *2.3B*. Celui à diode Zener est évidemment plus précis et assure un *reset* dès que l'alimentation descend en dessous de V_z + 0,7 V, tandis que celui sans diode Zener réalise la même chose pour une baisse en dessous de $0{,}7(1 + R_2/R_1)$.

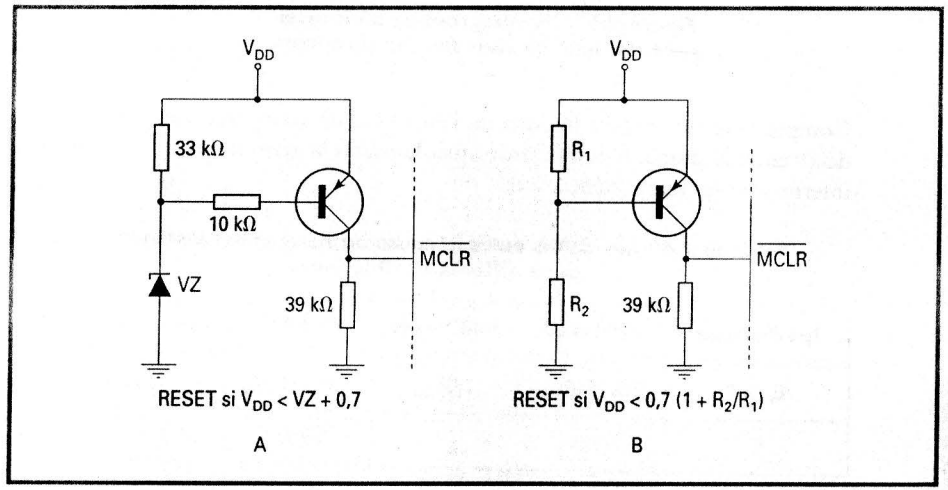
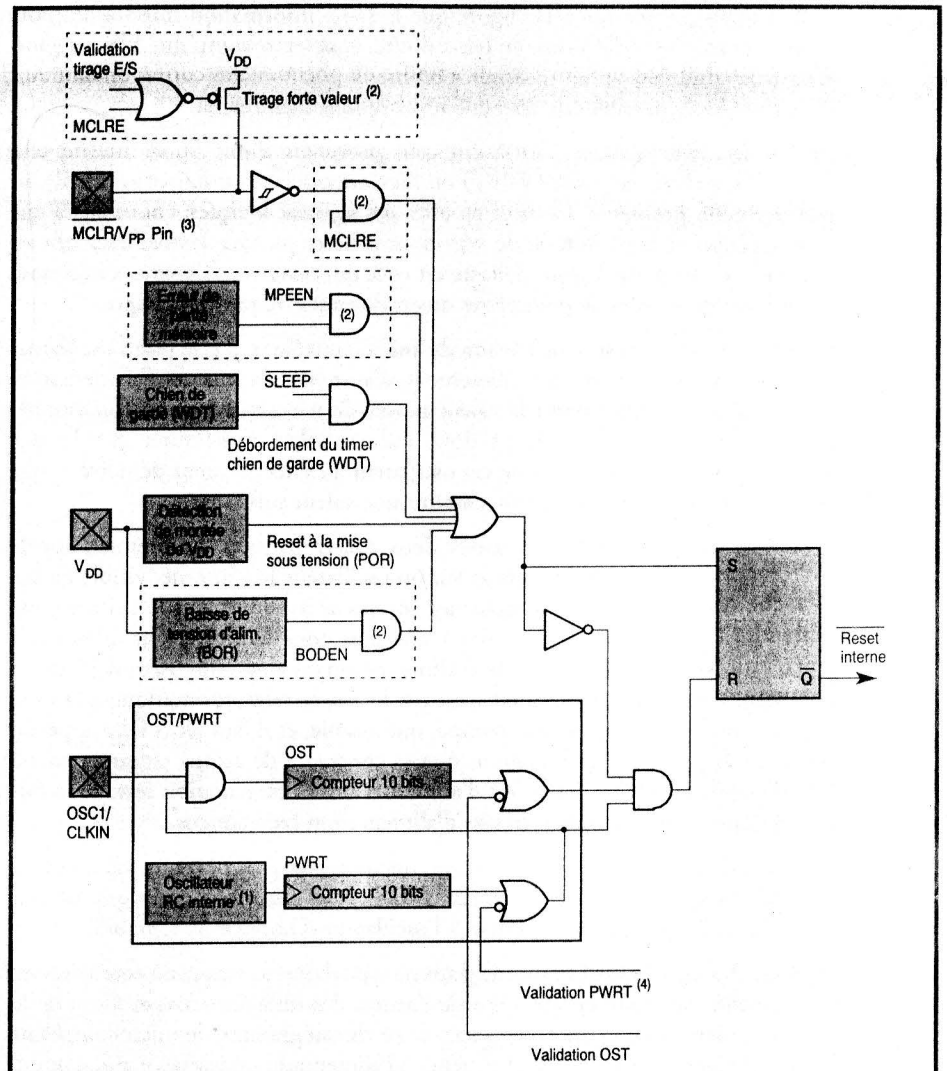


Figure 2.3 – Reset automatique en cas de baisse de la tension d'alimentation pour les circuits non munis du BOR.

2.2.3 Schéma interne de la circuiterie de *reset*

La *figure 2.4* vous propose de découvrir une partie du schéma interne de la circuiterie de *reset* des microcontrôleurs PIC, ce qui va nous permettre de comprendre comment se déroulent réellement les différentes séquences de *reset*.

© Dunod. La photocopie non autorisée est un délit.



Remarques :
1. Cet oscillateur est indépendant de tous les autres oscillateurs RC internes du circuit.
2. Les ressources entourées de pointillés ne sont pas disponibles dans tous les circuits.
3. Dans certains circuits, cette patte peut être configurée en entrée à usage général.
4. Les premiers PIC valident le PWRT avec le bit correspondant mis à 1, alors que les circuits les plus récents le valident avec le bit correspondant à 0.

Figure 2.4 – Schéma équivalent interne de la circuiterie de reset dans le cas le plus complet.

Tout d'abord, notez sur cette figure que la seule information importante pour l'unité centrale est celle située en bas à droite, appelée *reset* interne. Elle est générée par la combinaison d'un certain nombre de phénomènes correspondant aux diverses sources possibles de *reset* que nous venons d'énumérer.

En fait, les *reset*s logiques, c'est-à-dire ceux provenant d'une source interne telle que le *timer* chien de garde (WDT) ou bien encore l'erreur de parité (PER), ne posent aucun problème. Ce sont en effet des signaux logiques « normaux » qui sont appliqués à la circuiterie de *reset* pour générer presque immédiatement un *reset* interne. La partie la plus délicate est celle qui concerne le *reset* à la mise sous tension car de nombreux paramètres divers doivent être pris en compte.

En effet, lors de la mise sous tension du microcontrôleur, d'une part l'oscillateur d'horloge ne démarre pas immédiatement, d'autre part la tension d'alimentation n'atteint pas immédiatement la valeur nominale nécessaire au bon fonctionnement du microcontrôleur. Il est donc indispensable de s'assurer que le *reset* interne n'est généré qu'après que cet oscillateur ait effectivement démarré et que lorsque la tension d'alimentation a atteint une valeur suffisante.

Pour cela, les PIC mettent en œuvre deux *timers* distincts. Le premier, appelé PWRT pour *power up timer*, repose sur un oscillateur RC interne, visible en bas à gauche de la *figure 2.4* qui, grâce au compteur à 10 bits qui le suit, impose d'attendre typiquement 72 ms suite à une mise sous tension avant de pouvoir générer un *reset* interne. La tension d'alimentation est alors supposée avoir atteint un niveau stable et correct. Si tel n'est pas le cas, le *reset* automatique à la mise sous tension doit être considéré comme inutilisable, et il faut alors faire appel au circuit de la *figure 2.2* en s'assurant que sa constante de temps est supérieure à celle d'établissement de la tension d'alimentation. Cette situation reste toutefois assez exceptionnelle avec des circuits d'alimentation bien conçus.

Pour ce qui est de l'oscillateur d'horloge, un autre compteur 10 bits, validé par le bit OST, permet là encore d'attendre 1 024 cycles d'horloge avant que ne soit généré le *reset* interne, ce qui permet à l'oscillateur d'horloge de se stabiliser.

La *figure 2.5* montre ainsi le chronogramme type d'une séquence de *reset* à la mise sous tension avec mise en évidence de l'action des différents *timers*. Comme le montrent bien les deux premiers tracés de ce chronogramme, le microcontrôleur fonctionne ici avec sa patte $\overline{\text{MCLR}}$ reliée à l'alimentation positive ce qui, comme nous l'avons vu en *figure 2.2A*, est le mode de câblage normal à utiliser pour exploiter la fonction de *reset* automatique à la mise sous tension.

Les autres sources de *reset* visibles sur la *figure 2.4* sont plus classiques et ne font appel qu'à des fonctions logiques combinatoires destinées à assurer leur prise en compte normale. Ce sont :

- la patte de *reset* externe $\overline{\text{MCLR}}$;
- l'information en provenance du *timer* chien de garde (WDT) ;

© Dunod. La photocopie non autorisée est un délit.

- l'information d'erreur de parité PER (non disponible sur tous les circuits rappelons-le) ;
- l'information de chute de tension d'alimentation BOR (également non disponible sur tous les circuits).

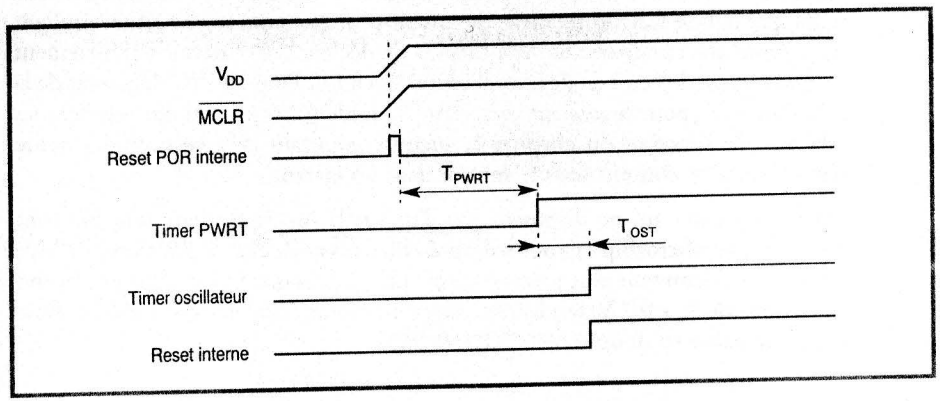


Figure 2.5 – Chronogramme d'un reset suite à une mise sous tension du circuit.

Compte tenu du schéma logique de l'ensemble de cette circuiterie de *reset*, les délais entre la situation devant provoquer le *reset* et la génération effective du *reset* interne sont indiqués *tableau 2.6*.

Tableau 2.6 – Les délais entre la cause du *reset* et le *reset* interne dans différentes situations.

Type d'oscillateur	PWRT validé	PWRT invalidé	*Reset* BOR	Sortie du mode sommeil
XT, HS, LP	72 ms + 1 024T_{OSC}	1 024T_{OSC}	72 ms + 1 024T_{OSC}	1 024T_{OSC}
RC	72 ms	-	72 ms	-

2.2.4 Le *reset* par détection de chute de tension d'alimentation ou BOR

Cette fonction très particulière confère une grande sécurité de fonctionnement aux microcontrôleurs qui en sont équipés. Elle permet en effet de générer automatiquement un *reset* lorsque la tension d'alimentation descend en dessous de la valeur minimum permettant de garantir le fonctionnement normal du microcontrôleur. Il est alors préférable de le ré-initialiser plutôt que de le laisser continuer à fonctionner au risque de se comporter de manière totalement imprévisible.

Si cette fonction est présente sur le PIC que vous avez retenu, aucun circuit n'est nécessaire pour sa mise en œuvre. Il suffit en effet de la valider au moyen du bit adéquat du registre PCON dont nous verrons le contenu dans un instant.

Lorsque cette fonction est validée, un *reset* de type BOR est automatiquement généré dès que la tension d'alimentation descend en dessous de la valeur BV_{DD} précisée dans la fiche technique de chaque circuit. Cette tension dépend en effet du type de circuit utilisé et de sa plage normale de tension d'alimentation. Cette descente doit également avoir lieu pendant un temps supérieur au minimum indiqué, là aussi, dans la fiche technique de chaque circuit concerné.

La *figure 2.6* montre comment se déroule une séquence de *reset* BOR en présence de trois types de diminution de tension d'alimentation différents. Remarquez le délai minimum de 72 ms présent entre le retour de la tension d'alimentation à sa valeur nominale et la génération du *reset* interne, afin d'être certain que tous les paramètres du circuit sont à nouveau stabilisés.

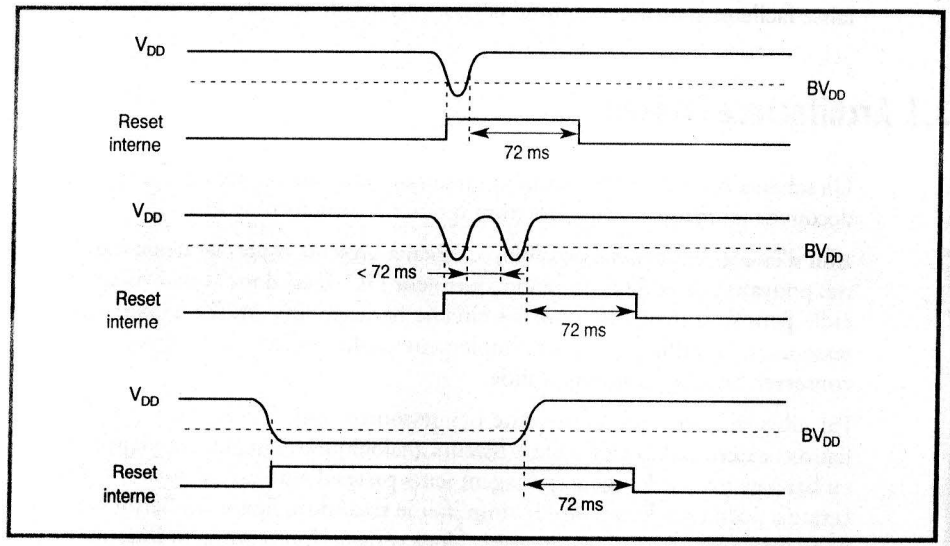


Figure 2.6 – Chronogramme d'un reset de type BOR pour trois types différents de diminution de VDD.

2.2.5 Les registres de contrôle et d'état du *reset*

Arrivé au terme de cette description des différents types de *reset*, il nous reste à vous présenter deux registres auxquels nous avons fait allusion à plusieurs reprises tout au long de cet exposé, dont certains bits permettent de savoir en présence de

© Dunod. La photocopie non autorisée est un délit.

quel type de *reset* l'on se trouve. Ce sont le registre PCON pour *power control register* et le registre STATUS ou registre d'état.

Voici tout d'abord la signification des bits du registre PCON qui comporte au maximum quatre bits utiles.

Accès E : Écriture L : Lecture	L	L	L	L	L	L/E	L/E	L/E
PCON	MPEEN	—	—	—	—	PER	POR	BOR
État à la mise sous tension	NA	0	0	0	0	0	0	0

- ***Bit 0 ou bit $\overline{BOR}$***

  Ce bit indique, lorsqu'il est à 0, qu'un *reset* dû à une détection de chute de la tension d'alimentation vient de se produire. Ce bit est un peu particulier, en ce sens que son état est inconnu suite à une mise sous tension du circuit. Il doit donc être positionné à 1 par programme. De même, après son passage à 0 suite à la détection d'une baisse de tension d'alimentation, il faut le remettre à 1 par programme.

- ***Bit 1 ou bit $\overline{POR}$***

  Ce bit indique, lorsqu'il est à 0, qu'un *reset* suite à une mise sous tension du circuit vient de se produire. Comme le bit 0, il faut le remettre à 1 par programme après son passage à 0 suite à la détection d'un *reset* de mise sous tension.

- ***Bit 2 ou bit $\overline{PER}$***

  Ce bit indique, lorsqu'il est à 0, qu'un *reset* suite à une erreur de parité de la mémoire de programme vient de se produire. Comme le bit 0, il faut le remettre à 1 par programme après son passage à 0 suite à la détection d'une erreur de parité mémoire.

- ***Bits 3 à 6***

  Ces bits n'existent pas et sont lus comme étant à 0.

- ***Bit 7 ou bit MPEEN***

  Ce bit recopie l'état du bit de configuration de la circuiterie de détection d'erreur de parité mémoire.

Le registre d'état ou STATUS, quant à lui, comporte divers bits affectés à l'indication d'état de l'unité centrale, que nous décrirons au chapitre suivant, et seulement deux bits qui, utilisés conjointement au contenu du registre PCON, permettent de déterminer la cause d'un *reset*.

Accès E : Écriture L : Lecture	L/E	L/E	L/E	L	L	L/E	L/E	L/E
STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C
État à la mise sous tension	0	0	0	1	1	X	X	X

- *Bit 3 ou bit $\overline{PD}$ pour Power Down*

  Ce bit est mis à 0 lors de l'exécution de l'instruction SLEEP. Il est mis à 1 suite à une mise sous tension ou à l'exécution d'une instruction CLRWDT.

- *Bit 4 ou bit $\overline{TO}$ pour Time Out*

  Ce bit est mis à 0 lorsqu'un débordement du *timer* chien de garde (WDT) se produit. Il est mis à 1 suite à une mise sous tension, l'exécution d'une instruction CLRWDT ou SLEEP.

Les autres bits du registre d'état ne concernent pas le *reset* et sont donc présentés au chapitre suivant lors de la description des registres de l'unité centrale.

CHAPITRE 3

ARCHITECTURE INTERNE

L'organisation interne des microcontrôleurs PIC diffère quelque peu de celle à laquelle vous êtes peut-être habitués avec les microcontrôleurs classiques. Cela résulte bien évidemment du fait que ces circuits ont une architecture de type Harvard, comme nous l'avons expliqué au chapitre 1, alors que la majorité des circuits classiques utilise une architecture de Von Neumann. Malgré cela, elle se laisse facilement analyser comme nous allons le voir sans plus tarder.

3.1 Architecture interne

Un schéma étant bien plus probant qu'un long discours, nous vous proposons de découvrir en *figure 3.1* l'organisation interne des microcontrôleurs PIC.

Afin d'être le plus général possible, ce schéma présente toutes les ressources internes pouvant exister dans un microcontrôleur PIC. Il est donc bien évident que sa taille peut se voir réduite pour les circuits ne disposant que d'une partie de ces ressources. Il suffit alors tout simplement de les effacer de la *figure 3.1* pour conserver un schéma interne valide.

Par ailleurs, notez que la majorité des ressources qui doivent faire appel à des liaisons externes (USART, convertisseurs analogiques/digitaux, etc.) représentées en bas à gauche de la figure, partagent leurs pattes d'entrées/sorties avec celles de certains ports parallèles. Afin de simplifier le tracé de la figure, les liaisons correspondantes n'ont pas été représentées. Nous verrons donc comment fonctionne ce partage au moment opportun lors de la présentation détaillée de ces ressources dans la suite de cet ouvrage.

L'examen fonctionnel de ce schéma peut être conduit de diverses façons et nous partirons quant à nous de la mémoire de programme. Cette dernière est organisée en mots de 14 bits (12 bits sur les plus « petits » PIC de la famille 12Cxxx que sont les 12C508, 509, 518 et 519), ce qui permet, comme nous l'avons vu au chapitre 1, de coder sur un seul mot l'instruction et l'opérande ou son adresse (au sens large du terme). Le code obtenu est ainsi très compact.

© Dunod. La photocopie non autorisée est un délit.

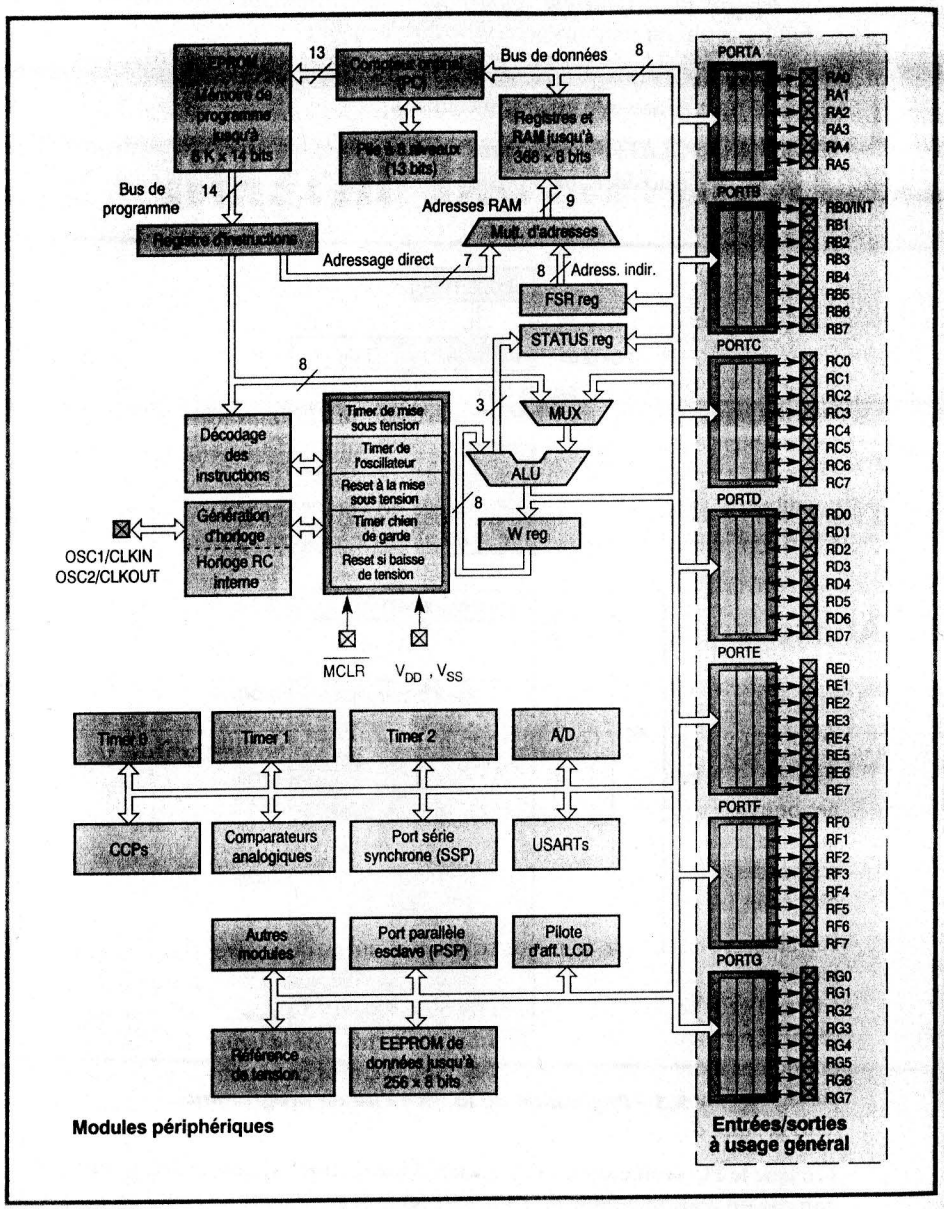


Figure 3.1 – Architecture interne des microcontrôleurs PIC.

Les données issues de cette mémoire entrent ensuite dans le registre d'instructions pour être dirigées sur le décodeur d'instructions. Ce dernier agit bien évidemment sur l'unité arithmétique et logique (ALU), ainsi que sur le registre W ou registre de travail (W pour *working register*). Ce dernier est intimement lié à l'ALU qui reçoit par ailleurs, via le bus de données spécifique (toujours à cause de l'architecture Harvard), les données issues du bloc de registres, des entrées/sorties ou des registres des ressources internes.

Ce bus de données aboutit donc sur le bloc de registres à usage général qui est à considérer en fait comme de la mémoire vive dont une partie des premiers octets est accessible directement. Ceux-ci correspondent aux registres spécifiques du circuit que nous découvrirons dans un instant.

Le compteur ordinal ou PC dispose d'une pile à huit niveaux pour les appels de sous-programmes ou le traitement des interruptions. Notez également la présence de deux registres situés à proximité de l'ALU. Le registre STATUS ou registre d'état, commun à tous les microprocesseurs, indique, au moyen de bits spécifiques, le « résultat » de certaines instructions exécutées par l'ALU (résultat nul, débordement, etc.). Le registre FSR quant à lui est propre aux PIC et permet de réaliser un adressage indexé un peu particulier comme nous le verrons dans un instant.

Au niveau de la circuiterie d'horloge, on retrouve bien évidemment les différentes sources de *reset* ainsi que les divers *timers* que nous vous avons présentés au chapitre précédent.

Le fonctionnement de tout cet ensemble est régi par l'horloge système. Comme le montre la *figure 3.2*, le signal délivré par l'oscillateur d'horloge est divisé par quatre afin de constituer l'horloge système et de générer en interne quatre cycles différents Q_1 à Q_4. Le PC est augmenté d'une unité à chaque cycle Q_1, alors que les instructions sont recherchées en mémoire de programme et mémorisées dans le registre d'instructions à chaque cycle Q_4. Ce diagramme met clairement en évidence le fait qu'une instruction s'exécute alors que la suivante est en cours de recherche, ce qui est le propre de la structure pipe-line des circuits à architecture RISC.

Cette présentation générale étant faite, nous vous proposons maintenant de découvrir l'organisation détaillée de la mémoire de programme et de la mémoire de travail, qui mettent toutes deux en œuvre des mécanismes de pagination, souvent jugés à tort comme étant complexes.

3.2 Organisation de la mémoire

L'organisation de la mémoire de données des PIC est un peu particulière car, d'une part elle contient tout à la fois les registres des ressources internes et des registres à usage général que l'on peut considérer comme de la RAM de travail, et

© Dunod. La photocopie non autorisée est un délit.

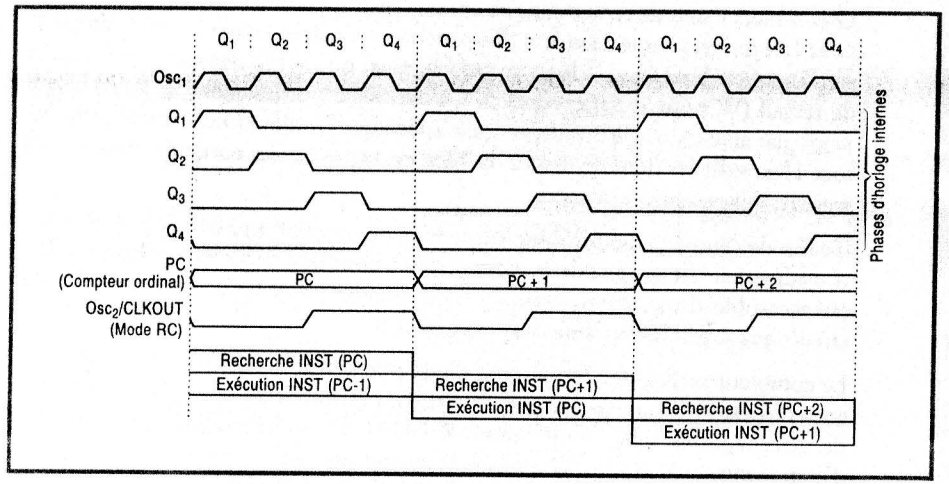


Figure 3.2 – Chronogramme d'exécution des instructions.

d'autre part elle fait appel à un mécanisme de pagination qui complique un peu son mode d'accès lorsque l'on n'est pas bien habitué à cette façon de faire.

La mémoire de programme s'étend linéairement de l'adresse 000 à l'adresse la plus élevée permise, compte tenu de la taille de la ROM ou de l'EEPROM intégrée dans le circuit choisi. Elle utilise aussi un mécanisme de pagination dont nous allons voir le principe sans plus tarder.

3.2.1 Mécanisme de pagination de la mémoire de programme

Comme le montre la *figure 3.3*, la mémoire de programme est divisée en plusieurs blocs de 2 Kmots de 14 bits (12 bits pour les plus petits PIC 12Cxxx) selon la taille de la mémoire totale disponible. Le vecteur de *reset* est toujours placé à l'adresse 000 tandis que le vecteur d'interruption se trouve à l'adresse 004.

Cela signifie que lors d'un *reset*, le compteur ordinal ou PC pointe toujours sur l'adresse 000 car le registre PCALTH (dont nous allons voir le rôle dans un instant) est toujours remis à 0.

Par contre, lors d'une interruption, le contenu de ce même registre PCLATH n'est pas modifié. Le PC pointe donc sur bien sûr l'adresse 004, mais il importe de tenir compte du fait que PCLATH n'a pas été modifié, si cela a une importance, dans le programme d'interruption.

Le compteur ordinal ou PC comporte 13 bits, ce qui lui permet d'adresser directement un maximum de 8 K de mémoire. Il est artificiellement divisé en trois registres distincts :

© Dunod. La photocopie non autorisée est un délit.

- le registre PCL (*program counter low*) qui correspond aux 8 bits de poids faibles et qui est toujours accessible directement ;
- le registre PCH (*program counter high*) qui correspond aux 5 bits de poids forts et qui n'est jamais accessible directement ;
- le registre PCLATH (*program counter latch high*) qui est l'intermédiaire obligé permettant d'accéder aux 5 bits de poids forts du PC contenus dans PCH.

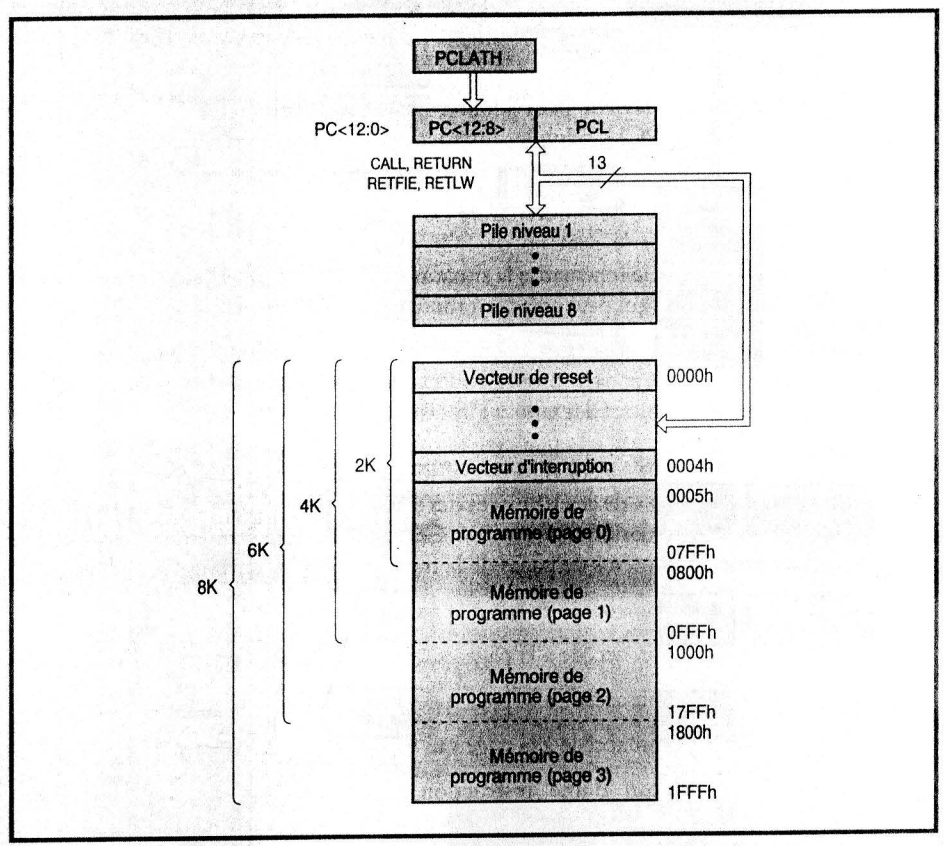


Figure 3.3 – Pagination de la mémoire de programme.

Lorsque le PC évolue suite à l'exécution d'instructions séquentielles, il fonctionne comme un registre unique de 13 bits de large et se rit donc de tout mécanisme de pagination mémoire.

Par contre, lors de l'exécution d'instructions de saut ou de branchement (GOTO, CALL et interruptions), il faut agir par programme sur le contenu de

PCLATH si un changement de page s'avère nécessaire, comme nous allons le voir en détail ci-dessous. Cela ne pose pas de problème théorique particulier, mais c'est une action que l'on oublie fréquemment au début et qui conduit alors à réaliser des sauts à des endroits totalement imprévus !

Le PC est, comme nous venons de le dire, un registre de 13 bits de large. Sa partie basse, constituée des 8 bits de poids faibles, est directement accessible en lecture et écriture au travers du registre appelé PCL (*program counter low*). Par contre, sa partie haute appelée PCH (*program counter high*) n'est pas directement accessible. Il faut passer pour cela par l'intermédiaire du registre PCLATH (*program counter latch high*).

Le contenu de ce registre peut adopter quatre comportements différents lors de la mise à jour de PCH, selon les instructions en cours d'exécution. La *figure 3.4* résume cela aussi bien qu'un long discours.

Lors d'une simple opération d'écriture dans PCL, le contenu réel du PC est obtenu en faisant précéder le contenu de PCL par les 5 premiers bits (bits 0 à 4) de PCLATH comme le schématise la situation 1 *figure 3.4*. On reconstitue bien ainsi un registre 13 bits qui peut accéder à toute la mémoire de programme disponible.

Lors d'une opération de saut au moyen d'une instruction GOTO, l'instruction contient dans son codage propre l'adresse de saut, mais sur 10 bits seulement. Cela permet de réaliser des sauts à l'intérieur d'une page de 2 K de la mémoire de programme, sans avoir à se préoccuper du mécanisme de pagination. Cependant, il faut savoir que la valeur du PC utilisée pour faire ce saut sera obtenue par la mise « bout à bout » de ces 10 bits et des bits 3 et 4 du registre PCLATH, comme indiqué sur la situation 2 *figure 3.4*. Ces bits peuvent être considérés comme constituant les numéros de pages de la mémoire de programme puisque :

- s'ils sont à 00, on accède en fait à la page qui va de 0 à 2 K ou page 0 ;
- s'ils sont à 01, on accède à la page qui va de 2 K à 4 K ou page 1 ;
- s'ils sont à 10, on accède à la page qui va de 4 K à 6 K ou page 2 ;
- s'ils sont à 11, on accède à la page qui va de 6 K à 8 K ou page 3.

Lors de l'exécution d'une instruction CALL, qui est un appel de sous-programme, le fonctionnement est identique puisque l'instruction véhicule aussi une adresse de saut sur 10 bits. On procède donc de la même façon que pour un GOTO afin de déterminer le contenu réel du PC. S'agissant d'un appel de sous-programme, le contenu du PC est bien évidemment sauvegardé sur la pile au préalable, comme le montre la situation 3 *figure 3.4*.

Lors de l'exécution d'une instruction de retour, que ce soit de sous-programme ou d'interruption, le PC est rechargé avec la valeur qui avait été placée sur la pile. Comme cette dernière a une largeur de 13 bits, aucune intervention du registre PCLATH ne s'avère nécessaire, comme le montre la situation 4 *figure 3.4*.

© Dunod. La photocopie non autorisée est un délit.

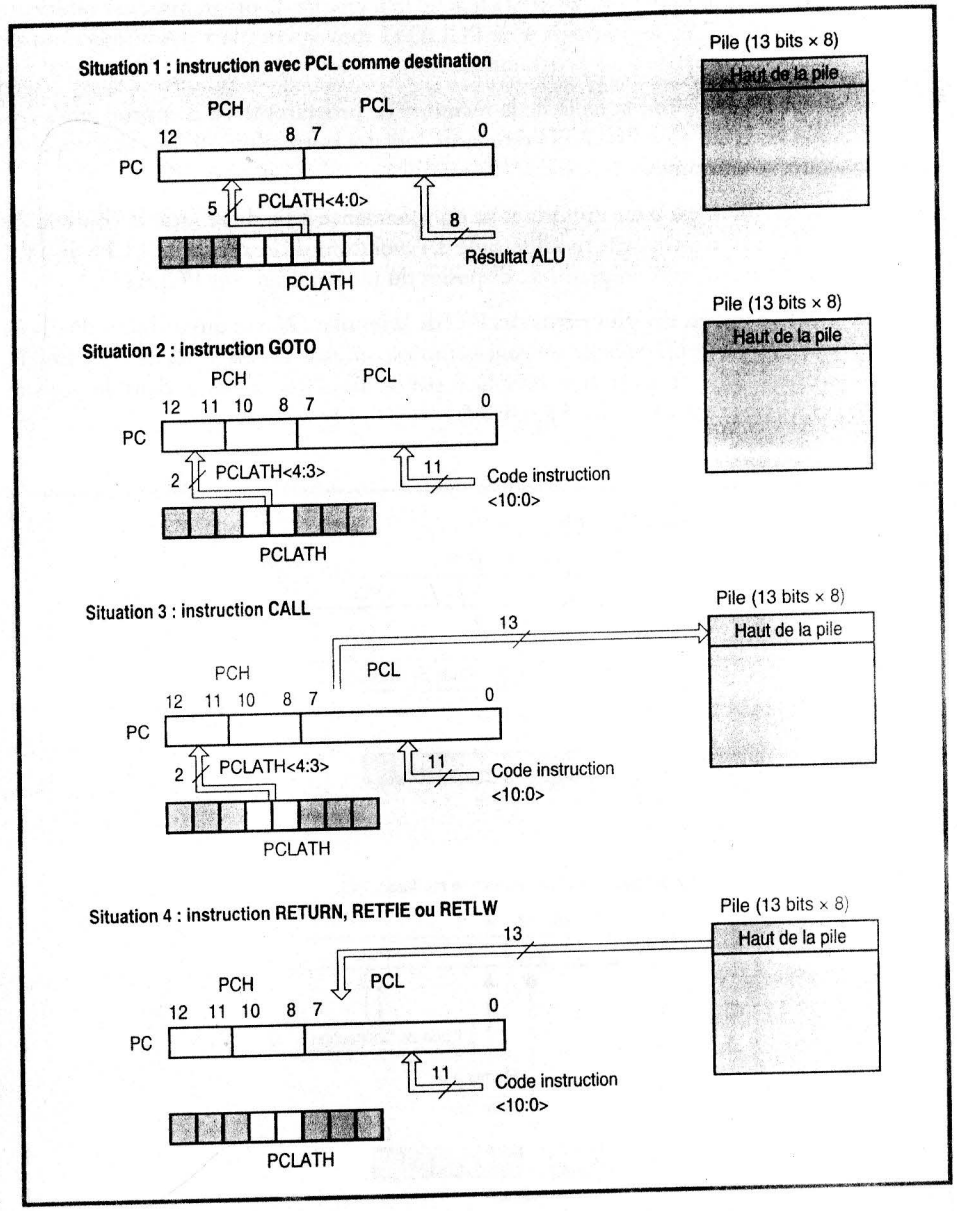


Figure 3.4 – Mécanisme de pagination de la mémoire de programme en fonction des instructions exécutées.

Pour ce qui est des PIC dont la taille de la mémoire de programme est inférieure ou égale à 2 K, les bits 3 et 4 de PCLATH sont ignorés lors des instructions de saut et d'appel de sous-programmes.

Pour les PIC dont la taille de la mémoire de programme est comprise entre 2 K et 4 K, le bit 4 de PCLATH est ignoré lors des instructions de saut et d'appel de sous-programmes.

Notez bien, car c'est important et on a tendance à l'oublier, que le contenu de PCLATH n'est jamais modifié suite à l'évolution du contenu de PCH. Il n'y a que vous qui, par programme, disposiez du moyen d'agir sur PCLATH.

Pour ce qui est des plus petits des PIC de la famille 12Cxxx qui utilisent des mots de 12 bits, le mécanisme de pagination est similaire, mais le registre PCLATH n'existe plus et se trouve remplacé par le bit PA0 contenu dans le registre STATUS, comme indiqué *figure 3.5*.

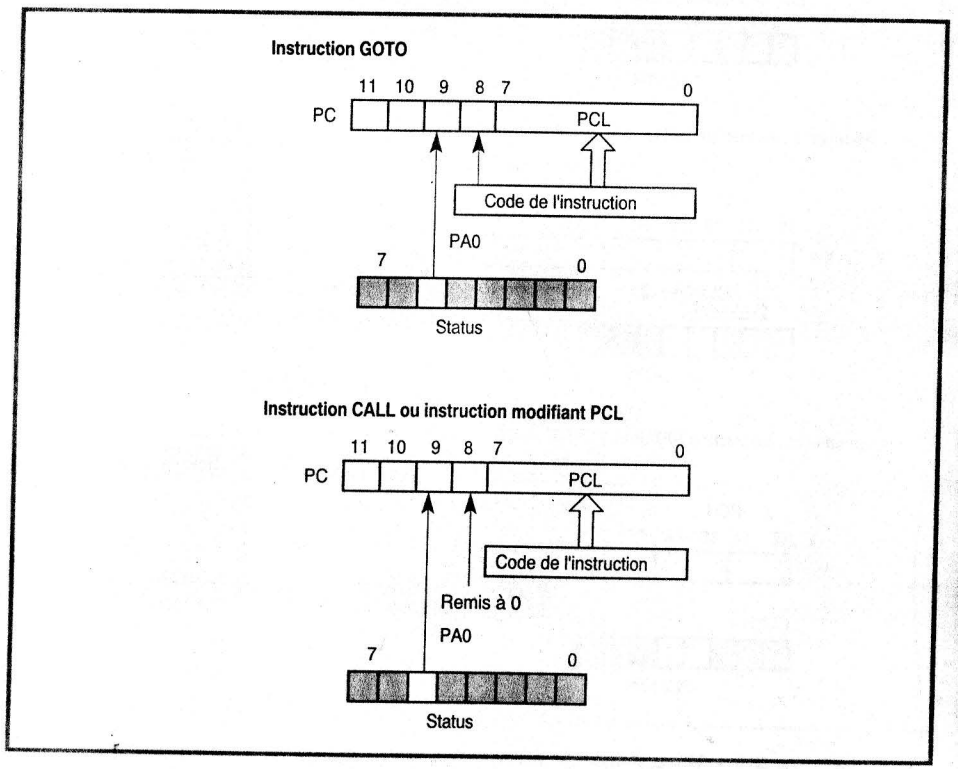


Figure 3.5 – Mécanisme de pagination de la mémoire de programme pour les PIC 12Cxxx à mots d'instructions de 12 bits.

© Dunod. La photocopie non autorisée est un délit.

3.2.2 Mécanisme de pagination de la mémoire de données

La mémoire de données des PIC, qu'il serait plus exact d'appeler le bloc de registres puisqu'elle contient tout à la fois les registres de l'unité centrale, ceux des ressources internes ainsi que des registres à usage général qui constituent en fait la véritable mémoire de données, utilise elle aussi un mécanisme de pagination.

Elle est divisée artificiellement en quatre pages d'une taille maximum de 128 octets chacune, cette taille et ce nombre de pages dépendant évidemment du circuit choisi.

Compte tenu de la taille maximum d'une page, qui est de 128 octets, il suffit de 7 bits pour adresser n'importe quel élément au sein d'une page. Comme le montre la *figure 3.6*, ces 7 bits font toujours partie intégrante du code de l'instruction.

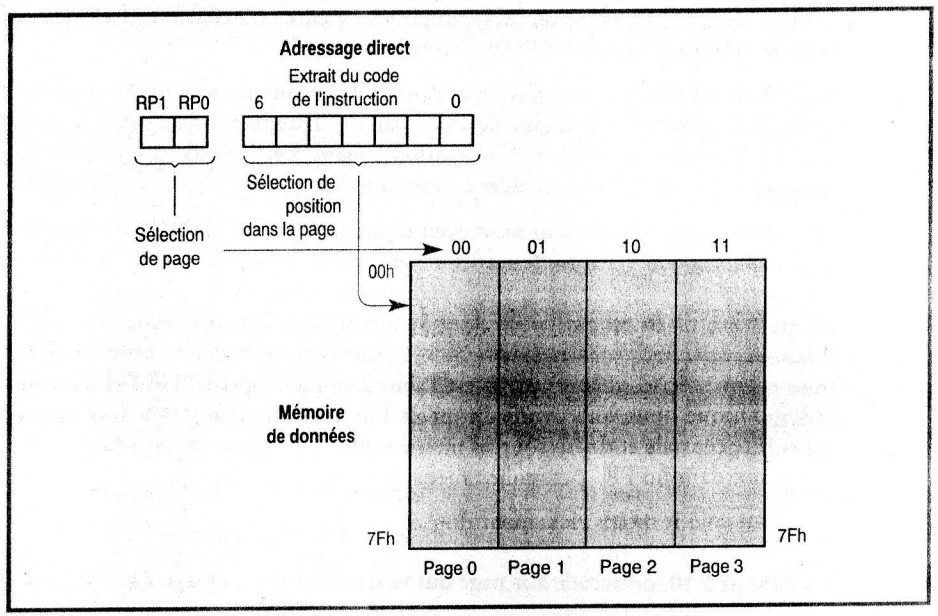


Figure 3.6 – Mécanisme de pagination de la mémoire de données en adressage direct.

Pour savoir quelle adresse complète sera réellement adressée par telle ou telle instruction, il faut en outre connaître le numéro de page mémoire utilisée, numéro qui est codé au moyen de deux bits appelés RP0 et RP1, contenus dans le registre d'état ou registre STATUS. En outre, lors de l'adressage indirect et comme le montre la *figure 3.7*, il faut également tenir compte de l'état du bit IRP contenu dans le registre F4 ou registre FSR.

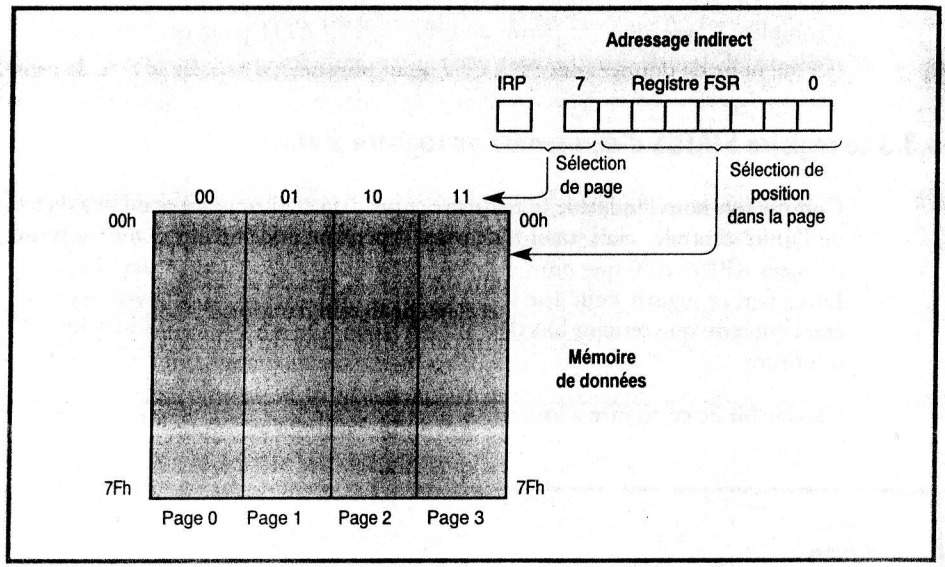


Figure 3.7 – Mécanisme de pagination de la mémoire de données en adressage indirect.

3.2.3 Organisation de la mémoire de données et des registres internes

Cela étant précisé, voyons quelle est l'affectation des adresses basses de chaque page, puisque nous vous avons dit qu'elles correspondaient en fait aux registres de l'unité centrale et des ressources internes. La *figure 3.8* vous propose un exemple choisi dans un PIC particulièrement bien pourvu en ressources.

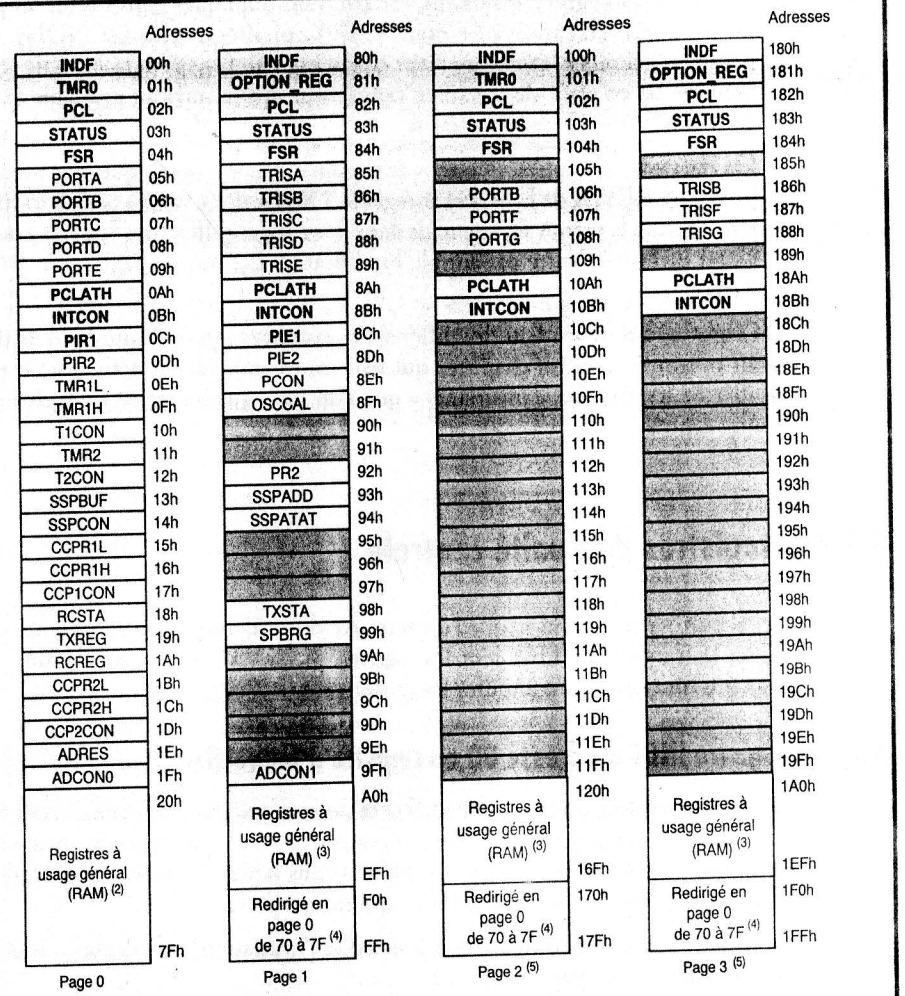
Dans un circuit moins richement doté, les registres qui correspondent à des ressources qui n'existent pas disparaissent de la figure. Les emplacements ainsi « libérés » ne sont cependant pas utilisables comme des registres à usage général et sont lus comme étant à 00.

L'examen de cette figure permet de noter un certain nombre d'informations importantes :

– Les registres à usage général, c'est-à-dire ceux que vous pouvez toujours utiliser comme mémoire vive de travail, se trouvent au-dessus de l'adresse 1F de chaque page (1F en page 0, 9F en page 1, 11F en page 2 et 19F en page 3).

– Un certain nombre de registres sont communs à toutes les versions de PIC. Ils sont toujours implantés aux mêmes adresses et ont leurs noms repérés en gras sur la *figure 3.8*. Ce sont les registres qui correspondent aux fonctions de base de l'unité centrale, que nous allons décrire dans un instant.

© Dunod. La photocopie non autorisée est un délit.



Remarques :
1. Les registres indiqués en gras sont présents dans tous les circuits.
2. Tous les registres ne sont pas présents dans tous les circuits. Les registres absents sont lus comme étant à 0 mais ne peuvent être utilisés pour un usage général.
3. Ces emplacements peuvent ne pas exister selon les circuits. Consulter la fiche technique du circuit concerné pour plus de précisions.
4. Certains circuits ne redirigent pas ces adresses en page 0.
5. Ces pages n'existent pas sur certains circuits.

Figure 3.8 – Adresses des principaux registres et organisation de la mémoire de données.

- Ces mêmes registres « invariants » se trouvent dupliqués dans toutes les pages (à une légère exception près pour TMR0 qui alterne avec OPTION). Cela permet d'y accéder immédiatement quelle que soit la page dans laquelle le programme est en train de travailler, sans devoir positionner au préalable les bits RP0 et RP1 mentionnés ci-dessus.
- Les registres « complémentaires » de certaines ressources tels que, par exemple, le registre RCSTA de la SCI et le registre TXSTA de ce même périphérique, se retrouvent à la même adresse mais dans deux pages différentes, afin de pouvoir passer de l'un à l'autre par simple modification des bits de pagination RP0 et RP1.

Les registres qui dépendent des différentes ressources internes sont bien évidemment présentés dans les chapitres qui leur sont consacrés ; il nous reste donc à étudier ici les registres « invariants » que l'on peut aussi appeler les registres de l'unité centrale.

3.3 Les registres de l'unité centrale

Ces registres « invariants », que l'on retrouve donc identiques à eux-mêmes dans toutes les versions de PIC, sont les registres propres à l'unité centrale ainsi que ceux qui sont communs à toutes les ressources internes.

3.3.1 Le registre INDF d'adresse 00 ou registre d'indirection

Le premier registre, ou registre d'adresse 00, est un peu particulier en ce sens qu'il n'a aucune existence physique. Vous ne pourrez donc pas y accéder. Si vous tentez de le lire, vous y trouverez toujours 00 et si vous tentez d'y écrire, cela produira un NOP, c'est-à-dire une instruction « qui ne fait rien ».

Ce registre sert en fait uniquement à spécifier l'utilisation de l'adressage indirect de la façon suivante par exemple :

```
ADDWF INDF,W
```

Cela signifie que l'on va ajouter au contenu du registre W le contenu du registre pointé par le registre FSR d'adresse 04. On réalise donc de l'adressage indirect par rapport au contenu de FSR en se servant de INDF comme mode de notation. C'est un peu particulier mais on s'y fait !

3.3.2 Le registre PCL d'adresse 02

Le registre d'adresse 02 quant à lui n'est autre que les 8 bits de poids faibles du PC comme nous l'avons expliqué lors de la présentation des mécanismes de pagination en début de chapitre. Rappelons que le contenu de ce registre est « complété » avec tout ou partie de celui de PCLATH pour générer la valeur réelle du PC lors de certaines instructions particulières.

3.3.3 Le registre STATUS d'adresse 03 ou registre d'état

Comme son nom l'indique, ce registre contient un certain nombre de bits d'état de l'unité centrale, mais aussi les bits de sélection de pages de la mémoire de données RP1 et RP0 que nous avons déjà évoqués à plusieurs reprises ci-dessus. De ce fait, ce registre peut être lu ou écrit comme n'importe quel autre registre, étant entendu que certains bits d'état resteront non modifiés suite à une tentative d'écriture.

Chaque bit de ce registre a une signification particulière que voici :

Accès E : Écriture L : Lecture	L/E	L/E	L/E	L	L	L/E	L/E	L/E
STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C
État à la mise sous tension	0	0	0	1	1	X	X	X

- *Bit 0 ou bit C pour carry*

  C'est le bit de retenue en addition ou soustraction. Il est positionné à 1 par les instructions ADDWF et ADDLW si une retenue est générée depuis le bit de poids fort, alors qu'il reste à 0 dans le cas contraire. Attention ! Pour les instructions de soustraction que sont SUBLW et SUBWF, il fonctionne « à l'envers » et se trouve donc à 1 en l'absence de retenue et à 0 en présence d'une retenue. Ce bit est également utilisé par les instructions de rotation (voir leurs descriptions respectives).

- *Bit 1 ou bit DC pour digit carry*

  C'est un bit de retenue « chiffre » pour l'arithmétique en DCB par exemple. Il est positionné à 1 lors des instructions ADDWF et ADDLW si une retenue est générée du bit 3 vers le quartet supérieur. Attention ! Comme le bit de carry, ce bit fonctionne « à l'envers » pour les instructions de soustraction que sont SUBLW et SUBWF ; il se trouve donc à 1 en l'absence de retenue et à 0 en présence d'une retenue.

- *Bit 2 ou bit Z pour zero*

  Ce bit est mis à 1 si le résultat de l'opération arithmétique ou logique exécutée est nul et à 0 dans le cas contraire.

- *Bit 3 ou bit $\overline{PD}$ pour power down*

  Comme nous l'avons vu au chapitre précédent, ce bit est mis à 0 lors de l'exécution de l'instruction SLEEP. Il est mis à 1 suite à une mise sous tension ou à l'exécution d'une instruction CLRWDT.

- *Bit 4 ou bit $\overline{TO}$ pour time out*

  Ce bit, déjà vu également au chapitre précédent, est mis à 0 lorsqu'un débordement du *timer* chien de garde (WDT) se produit. Il est mis à 1 suite à une mise sous tension ou à l'exécution d'une instruction CLRWDT ou SLEEP.

- *Bits 5 et 6 ou bits RP0 et RP1*

  Ces bits servent à sélectionner les pages de mémoire de données en mode d'adressage direct, comme nous l'avons vu ci-dessus lors de la présentation du mécanisme de pagination de la mémoire de données, et ceci conformément aux indications du *tableau 3.1.*

- *Bit 7 ou bit IRP*

  Ce bit sert lui aussi à sélectionner les pages de la mémoire de données, mais en mode d'adressage indirect comme nous l'avons vu ci-dessus lors de la présentation du mécanisme de pagination de cette dernière, et conformément aux indications du *tableau 3.2.* Il n'est pas présent sur tous les circuits de la famille puisqu'il est inutile sur ceux qui ne comportent que les deux premières pages de mémoire de données, c'est-à-dire sur les circuits avec moins de 256 octets de mémoire de données. Bien qu'il soit alors inutile, il doit cependant être laissé à 0 dans ces circuits.

Tableau 3.1 – Pagination de la mémoire de programme en adressage direct en fonction des bits RP0 et RP1 du registre STATUS.

RP1	RP0	Page	Adresse
0	0	0	00 à 7F
0	1	1	80 à FF
1	0	2	100 à 17F
1	1	3	180 à 1FF

Tableau 3.2 – Pagination de la mémoire de programme en adressage indirect en fonction du bit IRP du registre STATUS.

IRP	Page	Adresse
0	0 et 1	00 à FF
1	2 et 3	100 à 1FF

3.3.4 Le registre FSR d'adresse 04 ou registre de sélection de registre

Comme son nom l'indique et comme nous vous l'avons brièvement expliqué lors de la présentation du registre INDF, le contenu du FSR est utilisé en fait pour réaliser de l'adressage indirect. Ce registre contenant 8 bits, sa taille est suffisante pour les versions de PIC ne comportant que 2 pages de mémoire de données au maximum, puisque l'on atteint alors les 256 octets.

Cependant, afin d'être compatible avec les versions disposant d'une mémoire de données de taille plus étendue, une adresse sur 9 bits est systématiquement construite en utilisant le contenu de ce registre et le bit IRP du registre d'état comme cela a été vu *figure 3.7.* Cela justifie que nous vous ayons recommandé de laisser IRP à 0, même sur les circuits où il était à priori inutile.

3.3.5 Le registre PCLATH d'adresse 0A

Ce registre n'est cité que pour mémoire, puisque nous avons vu son rôle en détail ci-dessus lors de la présentation du mécanisme de pagination de la mémoire de programme.

3.3.6 Le registre PCON d'adresse 8E

Ce registre a déjà été traité au chapitre 2 lors de la présentation des différentes sources de *reset.* Rappelons rapidement son contenu.

Accès E : Écriture L : Lecture	L	L	L	L	L	L/E	L/E	L/E
PCON	MPEEN	—	—	—	—	$\overline{PER}$	$\overline{POR}$	$\overline{BOR}$
État à la mise sous tension	NA	0	0	0	0	0	0	0

- *Bit 0 ou bit $\overline{BOR}$*

  Ce bit indique, lorsqu'il est à 0, qu'un *reset* dû à une détection de chute de la tension d'alimentation vient de se produire. Ce bit est un peu particulier, en ce sens que son état est inconnu suite à une mise sous tension du circuit. Il doit donc être positionné à 1 par programme. De même, après son passage à 0 suite à la détection d'une baisse de tension d'alimentation, il faut le remettre à 1 par programme.

- *Bit 1 ou bit $\overline{POR}$*

  Ce bit indique, lorsqu'il est à 0, qu'un *reset* suite à une mise sous tension du circuit vient de se produire. Comme le bit 0, il faut le remettre à 1 par programme après son passage à 0 suite à la détection d'un *reset* de mise sous tension.

- *Bit 2 ou bit $\overline{PER}$*

  Ce bit indique, lorsqu'il est à 0, qu'un *reset* suite à une erreur de parité de la mémoire de programme vient de se produire. Comme le bit 0, il faut le remettre à 1 par programme après son passage à 0 suite à la détection d'une erreur de parité mémoire.

- *Bits 3 à 6*

  Ces bits n'existent pas et sont lus comme étant à 0.

- *Bit 7 ou bit MPEEN*

  Ce bit recopie l'état du bit de configuration de la circuiterie de détection d'erreur de parité mémoire.

3.3.7 Le registre OPTION d'adresse 81

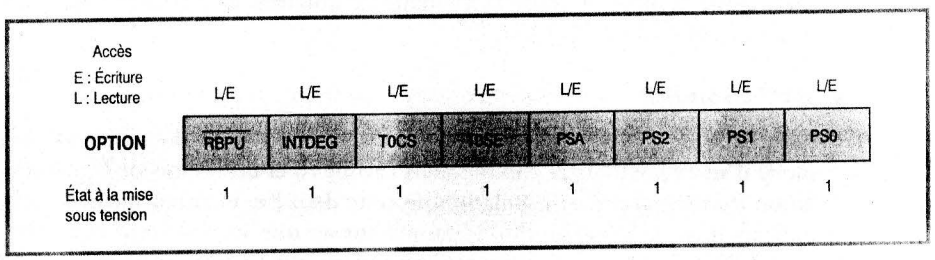
Ce registre est un petit peu un registre « fourre-tout » car il contient des bits de contrôle de diverses fonctions internes sans rapport entre elles. Nous le présentons à ce stade de l'ouvrage car il fait partie des registres « invariants » mais, comme vous allez le constater, il fait référence à diverses ressources internes et son contenu vous sera donc plus compréhensible lorsque vous aurez lu la description de ces dernières, au niveau desquelles d'ailleurs la description de ce registre sera rappelée.

- *Bits 0 à 2 ou bits PS0, PS1 et PS2 pour prescaler*

  Ces trois bits servent à définir le taux de prédivision appliqué à l'entrée du *timer 0* ou du *timer* chien de garde ou WDT conformément aux indications du *tableau 3.3*.

Tableau 3.3 – Taux de division du prédiviseur du *timer* 0 ou du *timer* chien de garde en fonction des bits PS0 à PS2 du registre OPTION.

PS2	PS1	PS0	*Timer 0*	Chien de garde
0	0	0	2	1
0	0	1	4	2
0	1	0	8	4
0	1	1	16	8
1	0	0	32	16
1	0	1	64	32
1	1	0	128	64
1	1	1	256	128



- *Bit 3 ou bit PSA pour prescaler assignment*

  Ce bit permet d'affecter le prédiviseur au *timer 0*, lorsqu'il est mis à 0 ou au *timer* chien de garde ou WDT lorsqu'il est mis à 1.

- *Bit 4 ou bit T0SE pour timer 0 source edge*

  Lorsqu'il est à 0, ce bit indique que le *timer 0* est incrémenté sur un front descendant de son signal d'entrée. Il est incrémenté sur un front montant lorsque ce bit est mis à 1.

- *Bit 5 ou bit T0CS pour timer 0 clock select*

  Lorsqu'il est à 0, ce bit sélectionne l'horloge instruction CLKOUT comme horloge du *timer 0*. Il sélectionne l'entrée T0CKI lorsqu'il est mis à 1.

- *Bit 6 ou bit INTEDG pour interrupt edge*

  Lorsqu'il est à 0, ce bit autorise la prise en compte d'une interruption provenant de la patte INT sur un front descendant. Il sélectionne un front montant lorsqu'il est à 1.

- *Bit 7 ou bit $\overline{RPU}$ pour RB port pull-up*

  Lorsqu'il est à 0, ce bit valide les résistances de tirage au niveau haut des entrées du port B. Il les invalide dans le cas contraire.

Un autre registre fait partie de la liste des registres « invariants » et devrait donc être présenté ici, c'est le registre INTCON chargé du contrôle des interruptions. Son contenu sera cependant plus compréhensible lorsque nous vous aurons présenté ces dernières, ce que nous allons faire tout de suite.

3.4 Les interruptions

Selon la version de PIC choisie, les sources d'interruptions peuvent être plus ou moins nombreuses. En effet, la majorité des ressources internes dispose d'une source d'interruption, voire parfois de deux selon le type de ressource.

Le principe général de traitement reste cependant identique pour tous les circuits, et seuls les registres de validation et d'autorisation des interruptions correspondantes augmentent au fur et à mesure que les ressources augmentent.

Dans le cas du circuit le mieux pourvu commercialisé à la date de rédaction de cet ouvrage, les sources d'interruptions possibles sont les suivantes :

- entrée d'interruption externe via la patte INT ;
- débordement du *timer 0* ;
- changement d'état d'une des pattes RB4 à RB7 du port parallèle B ;
- comparateur analogique ;
- port parallèle esclave ou PSP ;
- USART (émission et réception de données) ;
- convertisseur analogique/digital ;
- interface pour afficheurs à cristaux liquides ;
- écriture dans la mémoire EEPROM de données ;
- débordement du *timer 1* ;
- débordement du *timer 2* ;

- circuit de capture et de comparaison ou CCP ;
- port série synchrone ou SSP.

Sur les circuits les plus simples, un seul registre appelé INTCON pour INTerrupt CONtrol suffit à contrôler les différentes sources d'interruptions disponibles. Au fur et à mesure que les circuits « grossissent », des registres complémentaires lui sont ajoutés. Ils se nomment PIEx pour *peripheral interrupt enable* (registre d'autorisation d'interruption), et PIRx pour *peripheral interrupt register* (registre d'indication d'interruption). Compte tenu des circuits commercialisés actuellement, x peut varier de 1 à 3.

3.4.1 Principe de fonctionnement des interruptions

Pour ce qui est du traitement des interruptions, les microcontrôleurs PIC ne diffèrent pas ou très peu des microcontrôleurs classiques. Sous réserve qu'une interruption soit validée, au moyen de son bit individuel et/ou au moyen du bit de validation général, il se passe les opérations suivantes, dès sa prise en compte :

- le bit de validation général d'interruption GIE du registre INTCON (voir ci-dessous) est automatiquement mis à 0 afin d'interdire la prise en compte de toute nouvelle interruption ;
- le contenu du compteur ordinal ou PC est sauvegardé sur la pile ;
- le compteur ordinal ou PC est chargé par la valeur 004, ce qui signifie que la première instruction du programme de traitement de toutes les interruptions doit se trouver à cette adresse.

Compte tenu du fait que l'on ne dispose que du seul vecteur d'interruption d'adresse 004, il faut prévoir, au tout début du programme de traitement des interruptions, une séquence de test des registres INTCON et PIRx afin de déterminer quelle est la source de l'interruption. Notez à ce propos qu'un certain nombre de bits ou drapeaux d'indication d'interruption contenus dans les registres INTCON et PIRx doivent être remis à 0 par programme une fois qu'ils ont été testés, afin de pouvoir à nouveau être armés par les interruptions éventuelles suivantes. Cette nécessaire remise à 0 est précisée au niveau de la description des bits correspondants dans la suite de cet ouvrage.

Lors du retour d'interruption provoqué par l'instruction RETFIE, le bit GIE du registre INTCON est remis automatiquement à 1 afin d'autoriser à nouveau la prise en compte des interruptions.

Par rapport à certains microcontrôleurs classiques, il importe de noter que lors d'une interruption, le PC est le seul registre sauvegardé afin de disposer sur la pile de l'adresse de retour. Si vous avez besoin de sauvegarder d'autres registres (le registre d'état ou STATUS constitue généralement un minimum), vous devez le faire par programme au tout début du programme de traitement des interruptions.

Compte tenu du fait que la pile des PIC ne comporte que huit niveaux et que l'on ne sauvegarde que le PC à chaque fois, vous pouvez imbriquer jusqu'à huit interruptions et/ou appels de sous-programmes au maximum l'un dans l'autre. Si vous allez au delà, vous perdrez les adresses de retour les plus anciennes puisque cette pile fonctionne selon le principe du dernier entré-premier sorti (pile LIFO pour *last in first out*).

Cela étant précisé, la validation et l'indication des sources d'interruptions reposent, comme nous l'avons dit, sur le registre INTCON et sur les divers registres PIEx et PIRx dont nous allons maintenant détailler le contenu.

3.4.2 Le registre INTCON d'adresse 0B

Ce registre sert au contrôle global des interruptions et à l'indication de la provenance d'une partie d'entre-elles grâce à des bits d'état.

Accès E : Écriture L : Lecture	L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E
INTCON	GIE	PEIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF
État à la mise sous tension	0	0	0	0	0	0	0	0

- ***Bit 0 ou bit RBIF pour RB interrupt flag***

  S'il est mis à 1, ce bit indique un changement d'état de l'une des lignes RB4 à RB7 du port parallèle B (voir description de ce port au chapitre 6). Sur certains circuits ce bit se nomme GPIF mais cela ne change rien à sa fonction. Ce bit doit être remis à 0 par logiciel.

- ***Bit 1 ou bit INTF pour interrupt flag***

  S'il est mis à 1, ce bit indique une interruption provoquée par la ligne d'interruption externe INT. Ce bit doit être remis à 0 par logiciel.

- ***Bit 2 ou bit T0IF pour timer 0 interrupt flag***

  S'il est mis à 1, ce bit indique un débordement du *timer 0*. Ce bit doit être remis à 0 par logiciel.

- ***Bit 3 ou bit RBIE pour RB interrupt enable***

  S'il est mis à 1, ce bit autorise les interruptions provoquées par un changement d'état des lignes RB4 à RB7 du port parallèle B. Sur certains circuits, ce bit se nomme GPIE mais cela ne change rien à sa fonction.

- ***Bit 4 ou bit INTE pour interrupt enable***

  S'il est mis à 1, ce bit autorise les interruptions provoquées par la ligne d'interruption externe INT.

- ***Bit 5 ou bit T0IE pour timer 0 interrupt enable***

  S'il est mis à 1, ce bit autorise les interruptions dues au débordement du *timer 0*.

- ***Bit 6 ou bit PEIE pour peripheral interrupt enable***

  S'il est mis à 1, ce bit autorise toutes les interruptions d'origine périphérique, non masquées par leurs bits individuels de validation, à être prises en compte. S'il est mis à 0, toutes les interruptions périphériques sont masquées, même si leurs bits de validation individuels sont correctement positionnés.

- ***Bit 7 ou bit GIE pour global interrupt enable***

  S'il est mis à 1, ce bit autorise toutes les interruptions non masquées par leurs bits individuels de validation à être prises en compte. S'il est mis à 0, toutes les interruptions sont masquées, même si leurs bits de validation individuels sont correctement positionnés.

Comme c'est le cas sur quasiment tous les microcontrôleurs, les bits d'indication d'interruption individuels restent fonctionnels, même si les interruptions correspondantes ne sont pas autorisées.

Notez enfin que, sur les circuits ne contenant qu'un périphérique interne susceptible de générer des interruptions, le bit PEIE est remplacé par le bit de validation d'interruption émanant de ce seul périphérique.

3.4.3 Les registres PIEx

Comme nous l'avons expliqué ci-dessus, ces registres, numérotés à partir de un et dont le nombre maximum actuel est de trois, contiennent les bits individuels de validation des interruptions en provenance des diverses ressources internes.

Compte tenu du fait que leurs positions dans les différents registres PIEx dépendent du circuit concerné, vous ne trouverez ci-dessous que la liste de ces bits avec leurs appellations normalisées. La consultation de la fiche technique du circuit que vous aurez choisi vous permettra ensuite de placer ces divers bits aux emplacements adéquats de ses registres PIEx.

- *Bit TMR1IE pour timer 1 interrupt enable*

  S'il est mis à 1, ce bit autorise les interruptions dues au débordement du *timer 1*.

- *Bit TMR2IE pour timer 2 interrupt enable*

  S'il est mis à 1, ce bit autorise les interruptions dues à une égalité réussie au niveau du *timer 2.*

- *Bit CCP1IE pour CCP1 interrupt enable*

  S'il est mis à 1, ce bit autorise les interruptions en provenance du CCP1 ou système de capture et de comparaison 1.

- *Bit CCP2IE pour CCP2 interrupt enable*

  S'il est mis à 1, ce bit autorise les interruptions en provenance du CCP2 ou système de capture et de comparaison 2.

- *Bit SSPIE pour synchronous serial port interrupt enable*

  S'il est mis à 1, ce bit autorise les interruptions en provenance du port série synchrone ou SSP.

- *Bit TXIE pour transmit interrupt enable*

  S'il est mis à 1, ce bit autorise les interruptions en provenance du port série asynchrone lors d'une émission de donnée.

- *Bit RXIE pour receive interrupt enable*

  S'il est mis à 1, ce bit autorise les interruptions en provenance du port série asynchrone lors d'une réception de donnée.

- *Bit ADIE pour analog digital interrupt enable*

  S'il est mis à 1, ce bit autorise les interruptions en provenance du convertisseur analogique/digital.

- *Bit PSPIE pour parallel slave port interrupt enable*

  S'il est mis à 1, ce bit autorise les interruptions en provenance du port utilisé en mode esclave parallèle ou PSP.

- *Bit EEIE pour EEprom write complete interrupt enable*

  S'il est mis à 1, ce bit autorise les interruptions lors de la fin d'une écriture dans la mémoire EEPROM de données.

- *Bit LCDIE pour liquid crystal display interrupt enable*

  S'il est mis à 1, ce bit autorise les interruptions en provenance du port pour afficheurs à cristaux liquides.

© Dunod. La photocopie non autorisée est un délit.

- *Bit CMIE pour comparator interrupt enable*

  S'il est mis à 1, ce bit autorise les interruptions en provenance du comparateur analogique.

3.4.4 Les registres PIRx

Comme nous l'avons expliqué ci-dessus, ces registres, numérotés à partir de un et dont le nombre maximum actuel est de trois, contiennent les bits individuels d'indication des sources ou causes d'interruptions en provenance des diverses ressources internes.

Compte tenu du fait que leurs positions dans les différents registres PIR dépendent, comme pour les bits PIE, du circuit concerné, vous ne trouverez ci-dessous que la liste de ces bits avec leurs appellations normalisées.

- *Bit TMR1IF pour timer 1 interrupt flag*

  Ce bit est mis à 1 lors d'un débordement du *timer 1.* Il doit être remis à 0 par logiciel.

- *Bit TMR2IF pour timer 2 interrupt flag*

  Ce bit est mis à 1 lors de l'égalité du contenu du *timer 2* et de son registre de période (voir description du *timer 2* au chapitre 7). Ce bit doit être remis à 0 par logiciel.

- *Bit CCP1IF pour CCP1 interrupt flag*

  Ce bit a diverses significations, selon le mode de fonctionnement du système de capture et comparaison. En mode capture, ce bit est mis à 1 lorsqu'une capture a eu lieu. En mode comparaison, ce bit est mis à 1 lorsqu'une comparaison valide a eu lieu. En mode PWM, ce bit est inutilisé. Dans tous les cas, ce bit doit être remis à 0 par logiciel.

- *Bit CCP2IF pour CCP2 interrupt flag*

  Ce bit a diverses significations, selon le mode de fonctionnement du système de capture et comparaison. En mode capture, ce bit est mis à 1 lorsqu'une capture a eu lieu. En mode comparaison, ce bit est mis à 1 lorsqu'une comparaison valide a eu lieu. En mode PWM, ce bit est inutilisé. Dans tous les cas, ce bit doit être remis à 0 par logiciel.

- *Bit SSPIF pour synchronous serial port interrupt flag*

  Ce bit est mis à 1 lorsqu'une émission ou une réception est terminée sur le port série synchrone SSP.

- *Bit TXIF pour transmit interrupt flag*

  Ce bit est mis à 1 lorsque le registre d'émission du port série asynchrone est vide. Ce bit est automatiquement remis à 0 lorsqu'une donnée est écrite dans le registre d'émission TXREG.

- *Bit RXIF pour receive interrupt flag*

  Ce bit est mis à 1 lorsque le registre de réception du port série asynchrone est plein. Ce bit est automatiquement remis à 0 lorsqu'une donnée est lue dans le registre de réception RCREG.

- *Bit ADIF pour analog digital interrupt flag*

  Ce bit est mis à 1 lorsqu'une conversion analogique/digitale est terminée. Ce bit doit être remis à 0 par logiciel.

- *Bit PSPIF pour parallel slave port interrupt flag*

  Ce bit est mis à 1 lorsqu'une lecture ou une écriture a eu lieu sur le port utilisé en mode esclave parallèle. Ce bit doit être remis à 0 par logiciel.

- *Bit EEIF pour EEprom write complete interrupt flag*

  Ce bit est mis à 1 à la fin d'une écriture dans la mémoire EEPROM de données. Ce bit doit être remis à 0 par logiciel.

- *Bit LCDIF pour liquid crystal display interrupt flag*

  Ce bit est mis à 1 lors d'une interruption en provenance du port pour afficheurs à cristaux liquides. Ce bit doit être remis à 0 par logiciel.

- *Bit CMIF pour comparator interrupt flag*

  Ce bit est mis à 1 lorsque la sortie du comparateur analogique a changé d'état. Ce bit doit être remis à 0 par logiciel.

Comme pour le registre INTCON présenté ci-dessus, tous ces bits fonctionnent normalement, que les interruptions correspondantes soient validées ou non. De même, et sauf cas particulier dûment précisé ci-dessus, chaque bit doit être remis à zéro par le logiciel de traitement de l'interruption correspondante.

À ce stade de l'ouvrage, vous connaissez toutes les ressources de base des microcontrôleurs PIC. Il nous reste donc maintenant à aborder la présentation de celles qui sont spécifiques à chaque circuit, ce qui va être l'objet de chacun des chapitres suivants.

© Dunod. La photocopie non autorisée est un délit.

CHAPITRE 4

LA MÉMOIRE EEPROM DE DONNÉES

Au moment où ces lignes sont écrites, près d'une vingtaine de microcontrôleurs PIC disposent de mémoire EEPROM de données. Ce type de mémoire, qui se comporte presque comme de la mémoire vive mais avec la particularité de conserver son contenu pendant une durée indéterminée, même en l'absence d'alimentation, est extrêmement intéressant pour toutes les applications devant mémoriser des paramètres programmés par l'utilisateur d'une mise sous tension à une autre. Un exemple classique nous est fourni tous les jours avec la mémorisation des chaînes dans un récepteur TV ou un magnétoscope, mais c'est loin d'être le seul.

Cette mémoire EEPROM de données ne fait pas partie de l'espace adressable « normal » des PIC. Elle n'interfère donc en aucune manière, ni avec la mémoire de programme, ni avec les registres ou mémoire de données classique.

La taille de cette mémoire EEPROM de données varie de 16 octets à 256 octets selon les versions de circuits dans lesquels elle est implantée. Son adresse la plus basse part toujours de 00 et son adresse la plus haute est déterminée par sa taille : de 0F pour 16 octets à FF pour 256 octets.

Elle n'est accessible en lecture et en écriture qu'au travers de deux registres :

– un registre EEDATA d'adresse 08 pour les données ;

– un registre EEADR d'adresse 09 pour les adresses.

Deux registres de contrôle, EECON1 d'adresse 88 et EECON2 d'adresse 89, permettent quant à eux de définir le mode de fonctionnement de cette mémoire.

Cette EEPROM n'utilise aucune source de haute tension externe et fonctionne dans toute la plage de tension d'alimentation autorisée pour le PIC choisi. Sa durée typique de programmation est de 10 ms et sa durée de rétention des informations est celle de toute EEPROM de ce type, soit de plusieurs années. Elle est contrôlée par un *timer* interne qui lui est propre, vous déchargeant ainsi de toute mesure de temps par logiciel et n'empruntant aucune ressource aux *timers* normaux du circuit.

Le fait de vouloir écrire à une adresse déjà occupée efface automatiquement le contenu qui s'y trouve avant d'y placer la nouvelle donnée. De ce fait, aucune commande d'effacement spécifique n'est prévue. Il n'existe pas non plus de procédure d'initialisation suite à une mise sous tension, et le contenu de cette mémoire doit donc être considéré comme quelconque tant qu'il n'a pas été défini par votre programme.

Le principe d'utilisation de cette EEPROM en lecture est fort simple et respecte la chronologie suivante :

– écriture de l'adresse à lire dans le registre EEADR ;

– mise à un du bit RD du registre de contrôle EECON1 (voir ci-après) ;

– lecture de la donnée ainsi adressée dans le registre EEDATA.

Compte tenu de la vitesse de fonctionnement de cette mémoire, la donnée est disponible dans EEDATA dès la mise à 1 du bit RD. Il est donc possible de la lire par l'instruction qui suit immédiatement celle qui met ce bit à 1.

L'écriture dans cette mémoire, qui est en fait une programmation puisque nous sommes en présence d'une EEPROM, est un peu plus complexe pour d'évidentes raisons de sécurité. Elle doit se dérouler de la façon suivante :

– écriture de l'adresse où vous souhaitez écrire dans le registre EEADR ;

– écriture de la donnée dans le registre EEDATA ;

– exécution du programme type que voici, à l'exclusion de tout autre processus :

```
MOVLW 55
MOVWF EECON2
MOVLW AA
MOVWF EECON2
BSF EECON1,WR
```

Cette dernière instruction déclenche le processus d'écriture proprement dit. Lorsqu'il est terminé, le bit EEIF contenu dans le registre EECON1 est mis à 1 et, si l'interruption EEPROM a été autorisée via le bit EEIE du registre PIEx, une interruption est générée. Le bit EEIF doit être remis à 0 par logiciel, comme nous l'avons vu au chapitre précédent lors de sa description.

La procédure que nous venons de voir, et qui consiste à écrire 55 puis AA dans le registre EECON2, est une mesure de sécurité mise en place sur la puce afin d'éviter qu'un programme connaissant un moment « d'égarement » ne puisse modifier le contenu de la mémoire EEPROM par accident, en manipulant simplement les bits de EECON1.

Le registre de contrôle EECON1 contient un certain nombre de bits de contrôle dont voici la signification :

© Dunod. La photocopie non autorisée est un délit.

Accès E : Écriture L : Lecture	L	L	L	L/E	L/E	L/E	L/E	L/E
EECON1	–	–	–	EEIF	WRERR	WREN	WR	RD
État à la mise sous tension	0	0	0	1	1	X	0	X

- ***Bit 0 ou bit RD pour read data***

  Ce bit doit être mis à 1 pour lire une donnée. Il est automatiquement remis à 0 par le circuit lorsque la lecture de la donnée est effective.

- ***Bit 1 ou bit WR pour write data***

  Ce bit doit être mis à 1 pour écrire une donnée. Il est automatiquement remis à 0 par le circuit lorsque l'écriture de la donnée est effective.

- ***Bit 2 ou bit WREN pour write enable***

  Si ce bit est mis à 0, il interdit toute écriture dans l'EEPROM. Il doit être mis à 1 pour autoriser une écriture.

- ***Bit 3 ou bit WRERR pour write error***

  Ce bit est mis à 1 si une erreur se produit en écriture suite à un arrêt prématuré de celle-ci (*reset* ou action du *timer* chien de garde par exemple). Dans ces conditions, les contenus de EEDATA et EEADR sont inchangés pour permettre une reprise correcte de l'opération.

- ***Bit 4 ou bit EEIF pour EEPROM interrupt flag***

  Ce bit est mis à 1 lorsqu'une écriture est terminée. Ce bit doit être remis à 0 par logiciel.

- ***Bits 5 à 7***

  Ces bits sont inutilisés et lus comme étant à 0.

Le registre EECON2 quant à lui n'a aucune existence physique et il est donc impossible de le lire. Il ne sert que pour le processus de sécurisation en écriture en utilisant le petit programme présenté ci-dessus.

Malgré la très grande sûreté offerte par ce processus d'écriture, d'autres protections ont été mises en place pour éviter toute altération involontaire du contenu de la mémoire EEPROM de données. Ainsi, lors de la mise sous tension, le bit WREN est mis à 0 interdisant de ce fait toute écriture. De plus le *timer* de *reset* à

la mise sous tension d'une durée de 72 ms (voir chapitre 2 si nécessaire) inhibe également l'écriture dans l'EEPROM pendant toute la durée de son activation.

Lorsque le PIC dans lequel cette mémoire est implantée est protégé en lecture par programmation du bit approprié, comme nous l'avons vu au chapitre 1, le programme contenu en mémoire garde la possibilité de lire et d'écrire dans l'EEPROM de données. Par contre, toute opération de lecture ou d'écriture devient impossible depuis un programmateur externe.

CHAPITRE 5

LE *TIMER* CHIEN DE GARDE OU WDT ET LE MODE SOMMEIL

Ces deux fonctions, quoiqu'indépendantes, présentent de nombreux points communs. Il est donc logique de les traiter dans le même chapitre. Contrairement à de nombreuses autres ressources, elles sont présentes sur tous les circuits de la famille, du plus petit au plus grand, ce qui est parfaitement logique vu leur intérêt général.

5.1 Le *timer* chien de garde

Le *timer* chien de garde, ou WDT pour *watch dog timer*, est une fonction utilisée pour se prémunir des errements de certains logiciels ou des blocages pouvant se produire dans certaines situations. En effet, il est généralement constitué d'un compteur, incrémenté au rythme d'une horloge qui fonctionne quoi qu'il advienne et, lorsque ce compteur déborde, un *reset* est généré. Le logiciel qui s'exécute sur un circuit muni d'une telle ressource doit donc régulièrement venir mettre à zéro ce compteur afin d'éviter ce débordement fatal. Si le logiciel « se plante » ou si, pour une raison quelconque, il est empêché de venir effectuer cette mise à zéro, par exemple parce qu'il reste indéfiniment en attente dans une boucle, le *timer* chien de garde se charge de mettre fin à cette situation indésirable grâce au *reset* provoqué par son débordement.

Le *timer* chien de garde dont sont munis les microcontrôleurs PIC respecte à la lettre ce principe. Il est muni de son propre oscillateur autonome à cellule R-C interne. Il ne nécessite donc aucun composant externe et continue à fonctionner même lorsque l'horloge du PIC est arrêtée lors d'une instruction SLEEP de mise en sommeil par exemple.

Comme tous ses homologues, ce *timer* compte en permanence et le fait de déborder, c'est-à-dire d'arriver à FF, lui fait générer un *reset* du microcontrôleur. Si le PIC se trouve en mode sommeil lorsque ce débordement se produit, cela a pour effet de le faire sortir de ce mode comme nous le verrons ci-après.

© Dunod. La photocopie non autorisée est un délit.

Si vous ne souhaitez pas utiliser le *timer* chien de garde, il est possible de l'invalider par écriture d'un 0 dans un bit spécifique du circuit lors de sa programmation. Ce bit ne fait pas partie de l'espace adressable ni des registres classiques et vous ne le verrez donc sur aucune des figures présentant les contenus de ces derniers. Tous les programmateurs de PIC savent néanmoins y accéder et vous proposent donc de valider ou non le *timer* chien de garde (WDT dans la langue de Shakespeare rappelons-le).

La *figure 5.1* présente le schéma synoptique de ce *timer* chien de garde qui partage avec le *timer 0* un diviseur programmable, constitué d'un compteur 8 bits suivi par un multiplexeur, comme nous le verrons au chapitre 7.

Sans ce diviseur, le temps de débordement typique du *timer* est de 18 ms mais peut varier quelque peu avec la tension d'alimentation et la température. S'il n'est pas assez long, le prédiviseur du *timer 0* peut être affecté au *timer* chien de garde, où il sert alors de postdiviseur comme le montre la *figure 5.1*.

L'utilisation de ce diviseur et son taux de division sont définis au moyen des bits PSA et PS0 à PS2 du registre OPTION, comme nous l'avons vu au chapitre 3 et comme il est rappelé ci-dessous.

Accès E : Écriture L : Lecture	L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E
OPTION	RBPU	INTDEG	T0CS	T0SE	PSA	PS2	PS1	PS0
État à la mise sous tension	1	1	1	1	1	1	1	1

- ***Bits 0 à 2 ou bits PS0, PS1 et PS2 pour prescaler***

  Ces trois bits servent à définir le taux de division appliqué à l'entrée du *timer 0* ou au *timer* chien de garde conformément aux indications du *tableau 5.1*.

- ***Bit 3 ou bit PSA pour prescaler assignment***

  Ce bit permet d'affecter le prédiviseur au *timer 0* lorsqu'il est mis à 0 ou au *timer* chien de garde lorsqu'il est mis à 1.

Les autres bits du registre OPTION n'ont pas d'action sur le *timer* chien de garde.

Comme on peut le constater à l'examen du *tableau 5.1*, et avec le taux maximum de division permis qui est de 128, il devient possible d'obtenir des temps de débordement du *timer* chien de garde allant jusqu'à 2,5 s.

© Dunod. La photocopie non autorisée est un délit.

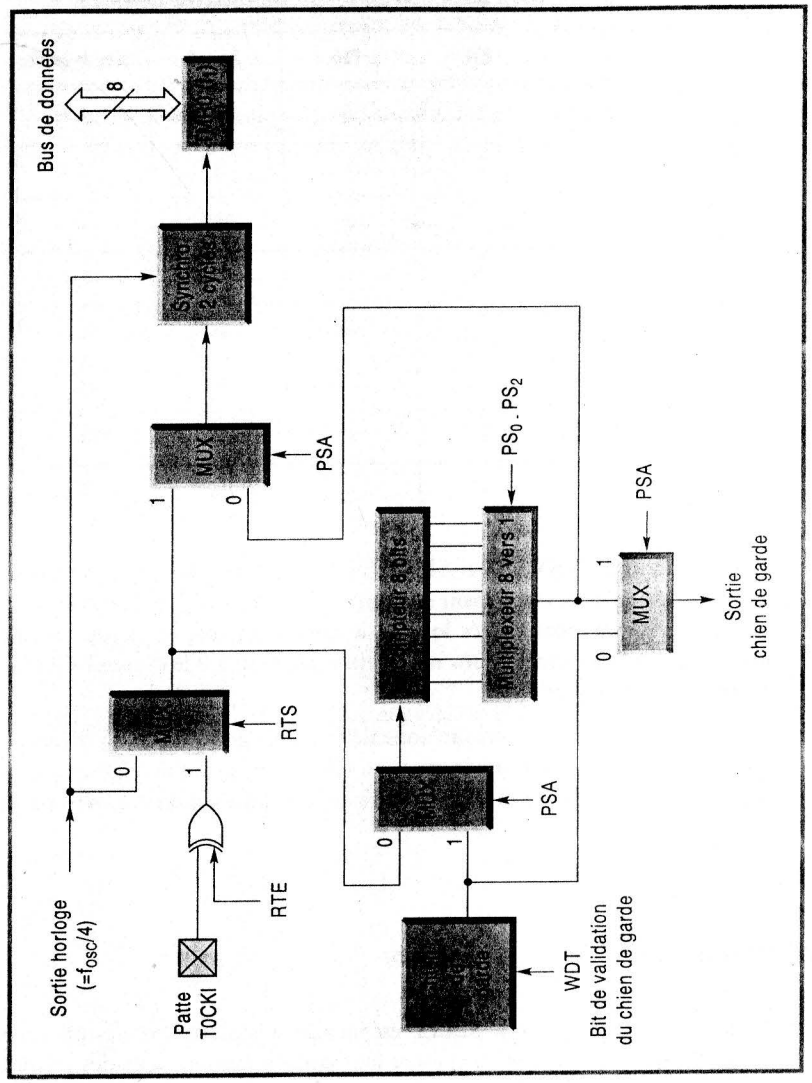


Figure 5.1 – Schéma synoptique du* timer *chien de garde ou WDT.

Tableau 5.1 – Taux de division du prédiviseur du *timer 0* ou du *timer* chien de garde en fonction des bits PS0 à PS2 du registre OPTION.

PS2	PS1	PS0	*Timer 0*	Chien de garde
0	0	0	2	1
0	0	1	4	2
0	1	0	8	4
0	1	1	16	8
1	0	0	32	16
1	0	1	64	32
1	1	0	128	64
1	1	1	256	128

Les instructions CLRWDT et SLEEP mettent à 0 le *timer* chien de garde et à 1 le bit TO du registre d'état (voir chapitre 2 si nécessaire). Cela permet d'éviter de générer des *reset* inconsidérés lorsque le circuit est mis en mode sommeil par exemple. Ces deux instructions remettent également à 0 le diviseur s'il est affecté au *timer* chien de garde.

Notez à ce propos que, lorsque vous désirez changer le taux de division, il est indispensable d'exécuter une instruction CLRWDT avant de réaliser cette modification car, dans le cas contraire, vous risquez de faire générer un *reset* parasite au *timer* chien de garde.

5.2 Le mode sommeil

Le mode sommeil ou mode SLEEP est un état très particulier des microcontrôleurs PIC qui leur permet de réduire leur consommation dans des proportions impressionnantes. En effet, lorsqu'ils sont dans cet état, l'oscillateur d'horloge système est arrêté et aucune activité n'a plus lieu au niveau du programme exécutable.

Contrairement à ce que vous pourriez penser au vu de cette courte présentation, ce mode est cependant très intéressant car certaines actions permettent de réveiller le PIC et donc de le faire sortir de cet état. Ainsi par exemple, il est possible de

réveiller un PIC au moyen d'un changement d'état sur certaines lignes de ses ports parallèles. Le mode sommeil peut donc être mis en œuvre dans une application qui attend une action sur un clavier pour réaliser telle ou telle fonction. Tant que le clavier n'est pas sollicité, le PIC reste en mode sommeil et sa consommation est alors dérisoire. Dès qu'une action sur le clavier a lieu, et sous réserve bien sûr que celui-ci soit connecté sur le port parallèle adéquat, le PIC est réveillé et se met à exécuter le programme correspondant. D'innombrables autres applications de ce mode sont permises car, comme nous allons le voir dans un instant, les causes de réveil sont assez nombreuses.

5.2.1 Particularités du mode sommeil

Lorsqu'il est mis en mode sommeil, le PIC remet à 0 le *timer* chien de garde mais continue ensuite à le laisser fonctionner normalement. Le bit PD du registre d'état (voir chapitre 2 si nécessaire) est mis à 0 et le bit TO de ce même registre est mis à 1. L'oscillateur d'horloge système est arrêté et tous les ports d'entrées/sorties conservent le niveau qu'ils avaient avant l'exécution de l'instruction SLEEP de mise en mode sommeil.

Afin de minimiser encore la consommation dans ce mode de fonctionnement, les entrées/sorties doivent être au niveau haut ou bas, c'est-à-dire à V_{DD} ou à V_{SS}, et toute consommation de courant sur ces dernières doit être évitée. Les entrées qui risquent de se trouver en haute impédance doivent être maintenues au niveau haut ou bas par une résistance de tirage, même de forte valeur. En effet, si elles restent flottantes, les changements d'états erratiques qui se produisent alors sous l'influence de perturbations extérieures induisent une augmentation de la consommation.

La patte de *reset* externe $\overline{\text{MCLR}}$ doit être maintenue au niveau logique haut. En outre, certains périphériques internes qui sont autorisés à fonctionner en mode sommeil comme nous le verrons dans les chapitres suivants, doivent être arrêtés préalablement au passage en mode sommeil si leur maintien en fonction n'est pas utile à l'application.

5.2.2 Le réveil ou la sortie du mode sommeil

La sortie du mode sommeil, appelée aussi comme le veut la logique « le réveil du circuit », peut être provoquée par trois types d'événements distincts :

- un *reset* du circuit ;
- un débordement du *timer* chien de garde si ce dernier avait été validé ;
- la validation d'un drapeau d'indication d'interruption par un des périphériques qui y sont autorisés en mode sommeil, et dont le bit d'autorisation de génération d'interruption contenu dans un des registres PIEx a été positionné.

© Dunod. La photocopie non autorisée est un délit.

Le premier type d'événement provoque évidemment un *reset* du circuit, analogue à celui qui se produirait si l'on avait actionné la commande de *reset* hors de tout mode sommeil. Le microcontrôleur reprend donc l'exécution du programme à son début.

Les deux autres types d'événements ont pour conséquence de faire reprendre l'exécution du programme à partir de l'instruction qui suit l'instruction SLEEP de mise en mode sommeil dans les conditions suivantes :

- Si le bit GIE de validation globale des interruptions en provenance des périphériques est mis à 0, c'est-à-dire si les interruptions sont interdites (voir chapitre 3 si nécessaire), le microcontrôleur exécute l'instruction qui suit l'instruction SLEEP puis continue ensuite normalement le programme en séquence.
- Si le bit GIE de validation globale des interruptions en provenance des périphériques est mis à 1, c'est-à-dire si les interruptions sont autorisées (voir chapitre 3 si nécessaire), le microcontrôleur exécute aussi l'instruction qui suit l'instruction SLEEP, mais il exécute ensuite le programme d'interruption. Si l'exécution de l'instruction qui suit l'instruction SLEEP n'est pas souhaitée dans ce cas, il faut impérativement que ce soit une instruction NOP (*No OPeration*) car il est impossible d'empêcher le PIC de l'exécuter. La seule solution possible consiste donc à la remplacer par une instruction « qui ne fait rien ».

5.2.3 Réveil au moyen des interruptions

Lorsque le bit GIE de validation globale des interruptions est mis à 0 mais que certaines interruptions individuelles sont validées par les bits correspondants des registres PIEx, ces dernières peuvent être utilisées pour réveiller le microcontrôleur comme nous venons de le voir. Il importe alors de bien comprendre ce qui se passe lorsque l'interruption se produit « au moment » de l'exécution de l'instruction SLEEP.

Si l'interruption se produit juste avant l'exécution de l'instruction SLEEP, celle-ci est alors automatiquement transformée en NOP, c'est-à-dire en instruction sans effet. Le programme continue alors à se dérouler en séquence comme si rien ne s'était passé.

Si l'interruption se produit pendant l'exécution de l'instruction SLEEP, l'instruction termine son exécution complète mais le circuit se réveille alors immédiatement après en respectant les règles présentées ci-dessus.

© Dunod. La photocopie non autorisée est un délit.

CHAPITRE 6

LES PORTS D'ENTRÉES/SORTIES PARALLÈLES

Les ports d'entrées/sorties parallèles font partie des ressources les plus utilisées sur les microcontrôleurs, toutes familles confondues. La facilité avec laquelle ils se programment et s'interfacent avec les éléments les plus variés est en grande partie responsable de ce succès.

Les microcontrôleurs PIC disposent de ports parallèles très variables en nombre, puisque ceux-ci sont nécessairement limités par la taille du boîtier et donc par le nombre de pattes disponibles. Tous fonctionnent selon le même principe mais tous n'adoptent pas la même circuiterie d'entrée ou de sortie. Nous allons donc voir tout d'abord les généralités relatives à ces ports et nous passerons ensuite en revue chaque type de port pouvant être proposé sur les différentes références de PIC avec, à chaque fois, présentation du schéma équivalent correspondant.

6.1 Généralités

Chaque port parallèle est contrôlé par deux registres : un registre appelé PORTX, ou registre de données, et un registre appelé TRISX ou registre de sens de fonctionnement du port. Actuellement, X peut être compris entre A et G, ce qui signifie que l'on peut avoir un maximum de 7 ports parallèles sur un seul et même boîtier, soit près de 50 lignes d'entrées/sorties parallèles (« près de » car tous ne font pas 8 bits de large !).

Les registres de données peuvent évidemment être lus et écrits selon que le port correspondant est utilisé en entrée ou en sortie. Par ailleurs, chaque ligne de chaque port peut être programmée individuellement en entrée ou en sortie selon un processus désormais classique sur quasiment tous les microcontrôleurs.

Lors d'un *reset*, et pour d'évidentes raisons de sécurité, tous les ports sont placés en entrée suite à l'initialisation de tous les registres TRIS correspondants.

La *figure 6.1* précise la structure de chaque ligne de port au moyen d'un schéma de principe général. Elle permet de comprendre l'essentiel des fonctionnalités de ces lignes, à savoir que les données en sortie sont « latchées » ou mémorisées si vous préférez, alors que les données en entrée ne le sont pas et sont donc lues « en temps réel ».

Cela conduit à prendre des précautions avec les instructions du type *read-modify-write* ou instructions du type lecture-modification-écriture comme par exemple BSF ou BCF, très utilisées pour tester les bits des ports. En effet, l'instruction de lecture lit réellement l'état de la patte du port concerné alors que l'instruction d'écriture écrit dans le latch correspondant ; il faut donc en tenir compte, notamment si les pattes ne peuvent pas réagir instantanément aux sollicitations qu'on leur impose, comme lorsqu'elles pilotent une forte charge capacitive par exemple.

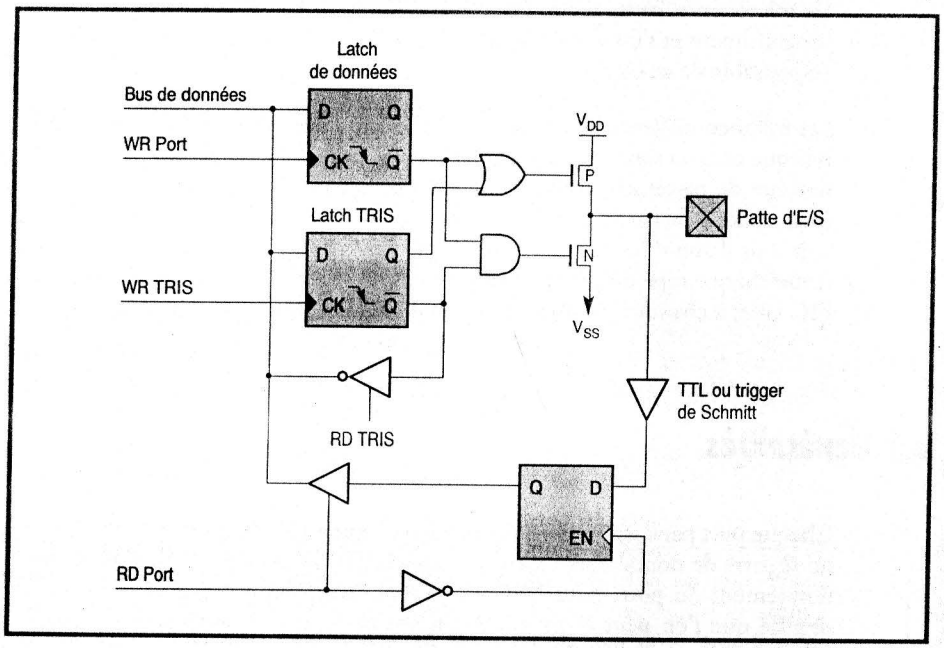


Figure 6.1 – Schéma équivalent général des entrées/sorties parallèles.

La *figure 6.1* permet aussi de constater que le sens de fonctionnement de chaque ligne de port est sous le contrôle du registre TRISX, où X est le nom du port concerné. Le bit n du registre TRIS correspond évidemment à la ligne n du port associé. Attention ! Un 0 programme la ligne correspondante en sortie alors qu'un 1 la programme en entrée, ce qui est l'inverse de ce que l'on rencontre dans nombre d'autres microcontrôleurs. Nos amis américains nous offrent d'ailleurs

© Dunod. La photocopie non autorisée est un délit.

gentiment un moyen mnémotechnique pour nous le rappeler en nous faisant remarquer qu'un 0 ressemble à un O comme Output et un 1 à un I comme Input. Hélas, transposé en français, ce n'est pas vraiment efficace…

Sur de nombreux circuits, certaines lignes des ports parallèles sont multiplexées ou partagées avec d'autres ressources internes. Cela influe parfois légèrement sur ce que nous venons de voir quant au comportement des ports et peut être résumé de la façon suivante : pour les pattes de ports parallèles qui sont partagées avec d'autres ressources, la validation de la ressource correspondante force les bits nécessaires dans le registre TRISx afin de placer ces pattes dans le sens nécessaire.

Les registres TRISx gardent cependant le contrôle du sens de fonctionnement des ports parallèles. Il est donc indispensable de s'assurer que les bits qui ont éventuellement été forcés ne changent pas d'état tant que la ressource correspondante est utilisée.

Les lignes de ports parallèles qui sont communes avec les entrées du convertisseur analogique/digital sont forcées en entrée suite à un *reset* de mise sous tension, conformément au contenu du registre ADCON1. La lecture de ces entrées au moyen du registre de port correspondant donne un 0 logique.

Les lignes de ports parallèles qui sont communes avec les entrées du comparateur analogique sont forcées en entrée suite à un *reset* de mise sous tension, conformément au contenu du registre CMCON. La lecture de ces entrées au moyen du registre de port correspondant donne un 0 logique.

Les lignes de ports parallèles qui sont communes avec les sorties de pilotage des segments des afficheurs LCD sont configurées en sortie suite à un *reset* de mise sous tension, conformément au contenu du registre LCDSE de cette ressource. Pour utiliser ce port comme un port parallèle numérique classique, les bits correspondants du registre LCDSE doivent être mis à 0. Tout bit mis à 1 dans le registre LCDSE est en effet prioritaire sur le bit correspondant du registre TRIS.

Lors de l'utilisation du port esclave parallèle (PSP) avec les ports D et E comme nous le verrons ci-après, vous devez vous assurer que le port D est bien configuré en entrée et que les lignes nécessaires du port E sont bien configurées en entrées/sorties numériques.

Avant d'en terminer avec ces généralités et pour changer un peu du logiciel, précisons que sur la majorité des boîtiers, chaque ligne de port prise individuellement peut fournir un courant maximal de 20 mA et absorber un courant maximal de 25 mA.

Attention ! Une limite à ces possibilités est imposée par les contraintes de dissipation de la puce qui font que le courant total fourni par chaque port ne peut dépasser 40 mA, alors que le courant total absorbé par chaque port ne peut dépasser 50 mA.

Tout cela étant précisé, voyons maintenant les particularités de chacun des ports parallèles que vous pouvez être amenés à rencontrer sur les différents PIC.

6.2 Le port A

Le port A dispose d'une largeur de six bits. Les lignes RA0 à RA3 d'une part et RA5 d'autre part respectent le synoptique interne visible *figure 6.2.*

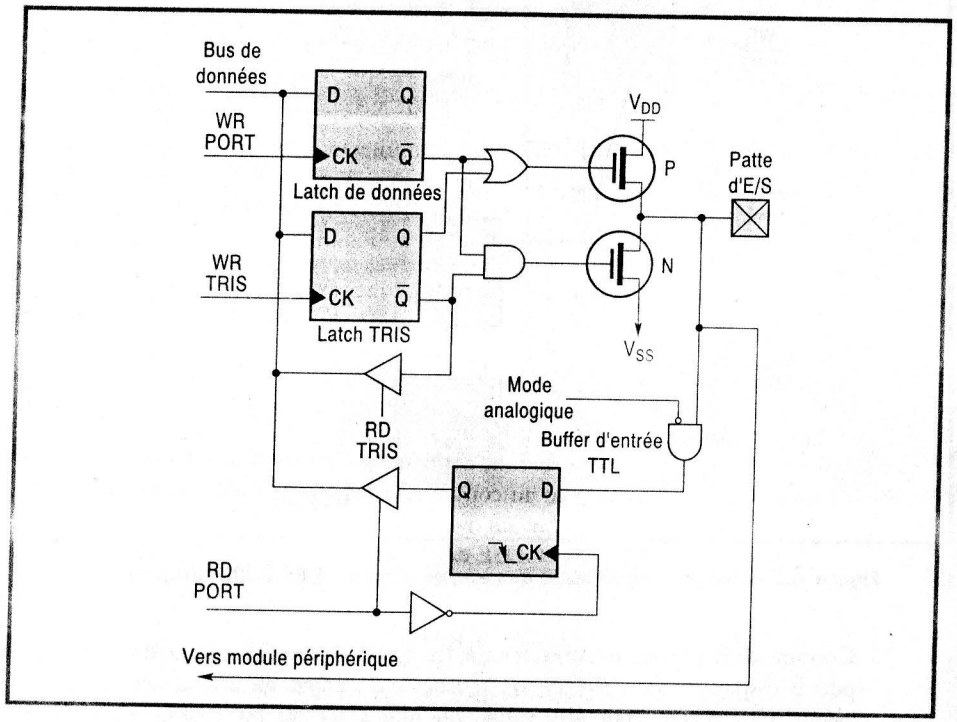


Figure 6.2 – Schéma équivalent des entrées/sorties parallèles RA0 à RA3 et RA5 du port A.

Comme le montre cette figure, elles sont toutes munies en sortie d'un « buffer » ou tampon CMOS trois états et acceptent en entrée des niveaux TTL.

Comme vous pouvez le constater sur cette même figure, les sorties sont mémorisées, ou « latchées » si vous préférez cet anglicisme. Si ces lignes sont partagées avec d'autres ressources, notez que ces dernières sont raccordées directement au niveau de la patte correspondante du boîtier comme le confirme bien ce schéma.

© Dunod. La photocopie non autorisée est un délit.

La ligne RA4 quant à elle adopte une structure un peu différente comme le montre le schéma de la *figure 6.3.*

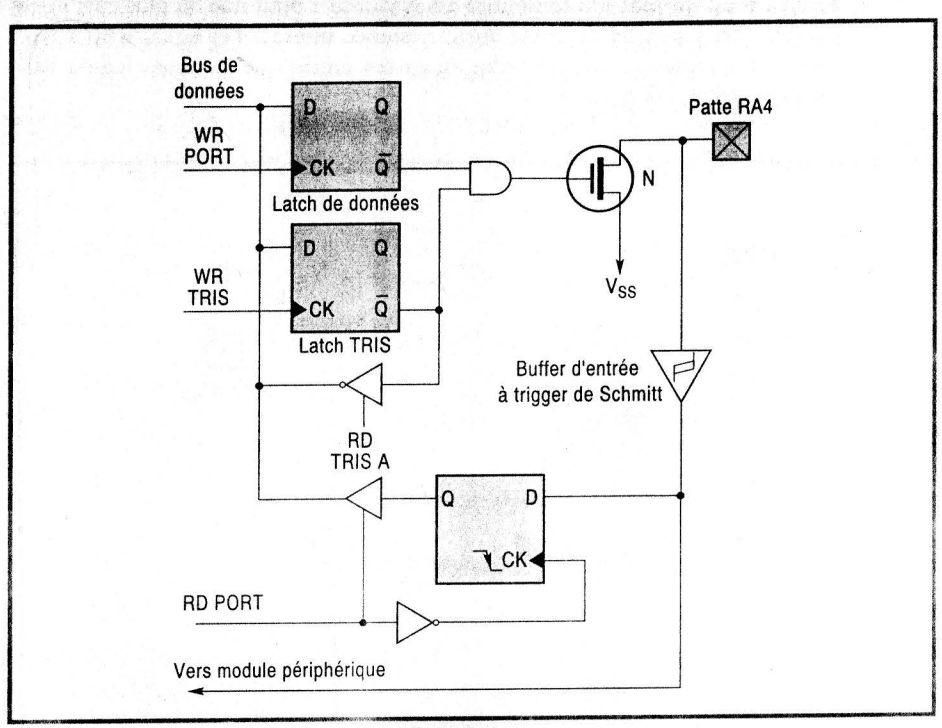


Figure 6.3 – Schéma équivalent de l'entrée/sortie parallèle RA4 du port A.

En effet, la sortie est cette fois-ci de type drain ouvert (l'équivalent en MOS du classique collecteur ouvert) et l'entrée est munie d'un trigger de Schmitt. Elle est généralement commune avec l'entrée externe du *timer 0* qui, dans ce cas, se trouve connectée après le trigger de Schmitt d'entrée.

Le sens de travail de toutes les lignes du port A est contrôlé par le registre TRISA, dans lequel un bit à 0 place la ligne correspondante en sortie et un bit à 1 la place en entrée. Suite à un *reset,* tous les bits du registre TRISA sont évidemment mis à 1.

6.3 Le port B

Le port B est un port bidirectionnel 8 bits complet dont une ou plusieurs lignes peuvent être partagées avec une autre ressource interne. Les lignes RB0 à RB3 adoptent la structure interne visible *figure 6.4* tandis que les lignes RB4 à RB7 adoptent celle visible *figure 6.5*.

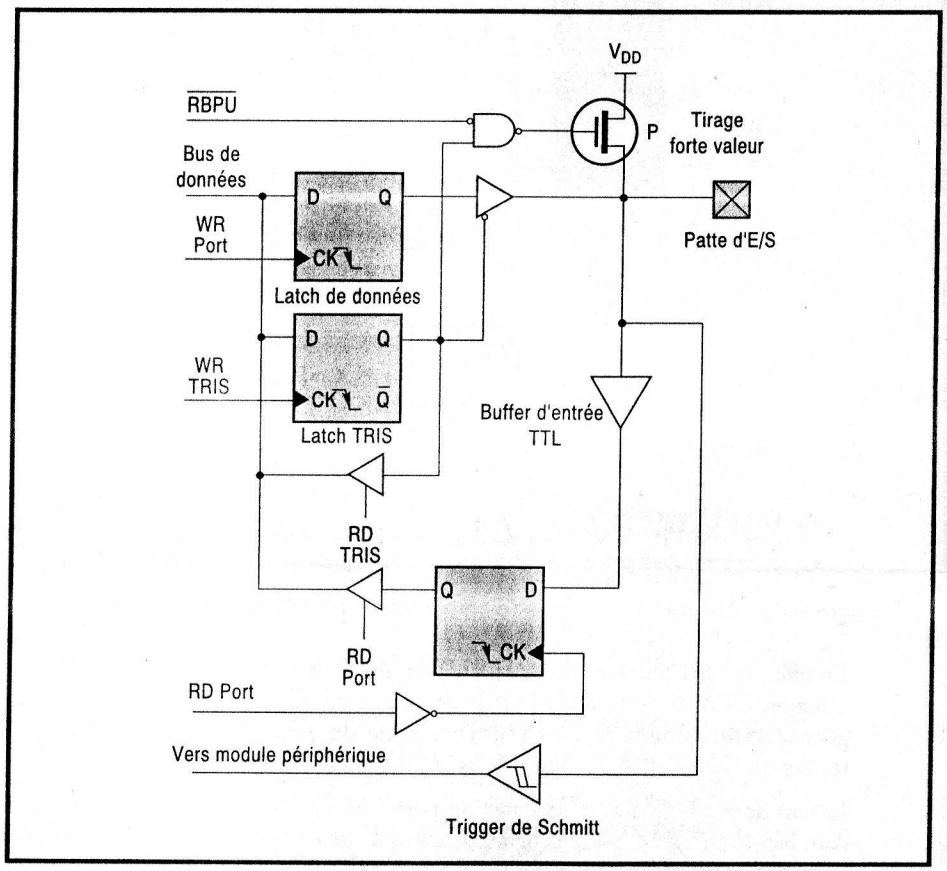


Figure 6.4 – Schéma équivalent des entrées/sorties RB0 à RB3 du port B.

La raison d'être de cette différence tient au fait qu'il est possible de programmer la génération d'une interruption lors d'un changement d'état de l'une quelconque des lignes RB4 à RB7, comme nous le verrons dans un instant.

© Dunod. La photocopie non autorisée est un délit.

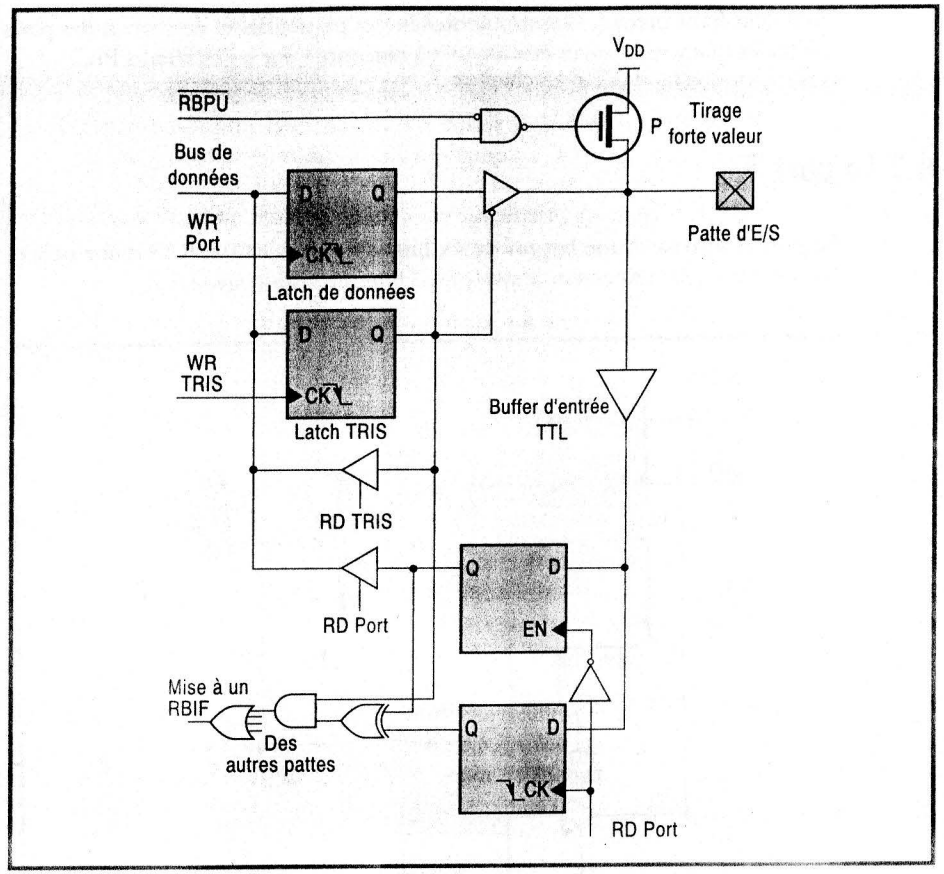


Figure 6.5 – Schéma équivalent des entrées/sorties RB4 à RB7 du port B.

Comme vous pouvez le constater sur les *figures 6.4* et *6.5*, toutes les lignes du port B disposent d'un *pull-up*, ou résistance de tirage, au niveau haut de forte valeur. Ce *pull-up* peut être validé ou non grâce au bit RBPU du registre OPTION (revoir si nécessaire la description complète de ce registre au chapitre 3). La validation ou non de ces *pull-up* est commune à toutes les lignes du port B. En outre, ils sont automatiquement désactivés lors d'un *reset* ainsi que pour celles des lignes du port qui sont configurées en sorties.

Le sens de travail de toutes les lignes de ce port est contrôlé par le registre TRISB dans lequel un bit à 0 place la ligne correspondante en sortie et un bit à 1 la place en entrée. Suite à un *reset*, tous les bits du registre TRISB sont évidemment mis à 1.

La possibilité de générer une interruption sur un changement d'état des lignes RB4 à RB7 du port B peut être validée au moyen du bit RBIF du registre INTCON décrit en détail au chapitre 3.

Si cette possibilité est utilisée, le simple fait qu'une quelconque des lignes RB4 à RB7 change d'état générera automatiquement une interruption grâce au principe suivant. Lors de la lecture de l'état des lignes RB4 à RB7, les valeurs lues sont mémorisées dans un latch interne. Le contenu de ce latch est ensuite comparé en permanence avec les valeurs présentes sur RB4 à RB7, et tout changement de l'une d'entre-elles provoque une interruption.

Cette interruption a la possibilité de réveiller le circuit placé en mode SLEEP, ce qui peut s'avérer très intéressant pour réaliser par exemple de la scrutation de clavier, le micro restant en veille, et donc consommant très peu, tant qu'aucune touche n'est activée.

Lors de l'exécution du programme de traitement d'interruption correspondant, la remise à zéro de l'interruption doit être réalisée en deux temps et dans l'ordre suivant :

- lecture ou écriture du port B afin d'annuler automatiquement la différence entre l'état des pattes et celui des latches internes ;
- remise à 0 du bit RBIF.

Attention ! Seules celles des lignes RB4 à RB7 qui sont programmées en entrées sont à même d'être surveillées par ce processus. Par ailleurs, notez qu'il est déconseillé de réaliser du *polling*, c'est-à-dire de la lecture à intervalles réguliers, du port B s'il est utilisé dans ce mode particulier. Cette lecture risque en effet d'annuler une différence d'état des lignes avant que l'interruption correspondante ait pu être déclenchée.

6.4 Le port C

Le port C est un port bidirectionnel 8 bits complet dont tout ou partie des lignes sont partagées avec d'autres ressources internes. Malgré cela, toutes les lignes de ce port adoptent la même structure visible *figure 6.6*.

Comme vous pouvez le constater sur cette figure, on dispose d'un buffer CMOS classique en sortie et d'un trigger de Schmitt en entrée.

Le sens de travail de toutes les lignes de ce port est contrôlé par le registre TRISC dans lequel un bit à 0 place la ligne correspondante en sortie et un bit à 1 la place en entrée. Suite à un *reset*, tous les bits du registre TRISC sont évidemment mis à 1.

© Dunod. La photocopie non autorisée est un délit.

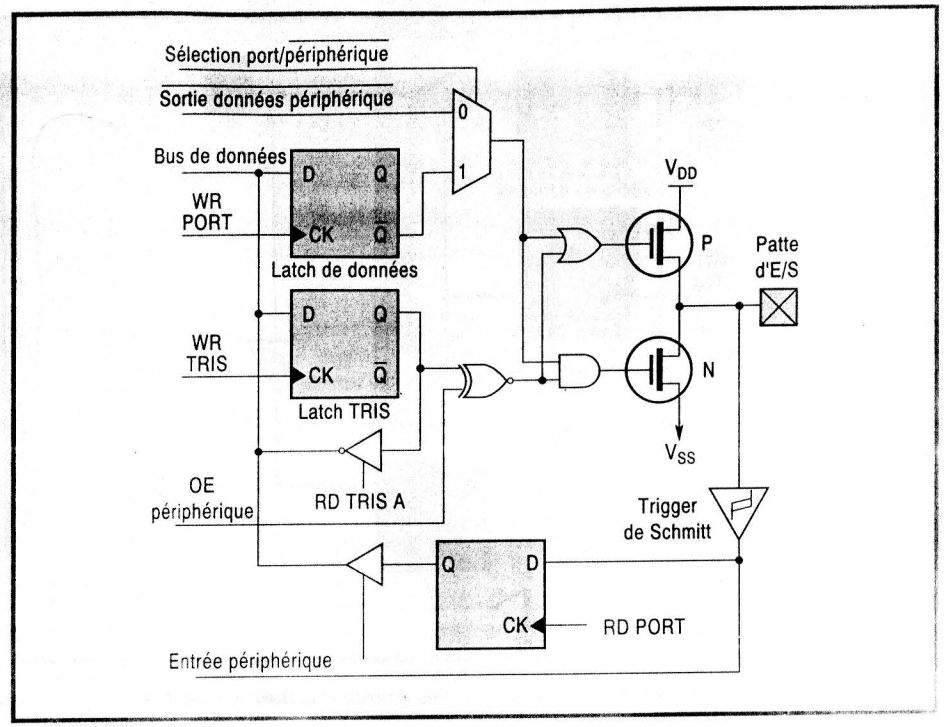


Figure 6.6 – Schéma équivalent des entrées/sorties du port C.

Par ailleurs, on note en haut et à gauche de la figure, la présence d'un multiplexeur qui permet d'envoyer sur la patte du port soit les données parallèles provenant du latch classique, soit les données provenant de la ressource avec laquelle le port C partage ses lignes. Le signal OE périphérique quant à lui, qui valide le buffer de sortie, n'est actif que si le multiplexeur est commuté du côté de la ressource interne. Dans le cas contraire, c'est-à-dire lorsque le port C fonctionne normalement, ce buffer est en effet contrôlé par le registre TRISC.

6.5 Le port D

Le port D est un port bidirectionnel 8 bits complet qui ne partage ses lignes d'entrées/sorties avec aucune autre ressource interne. Son schéma interne est visible *figure 6.7* sur laquelle on constate la présence d'un buffer TTL en sortie et d'un trigger de Schmitt en entrée.

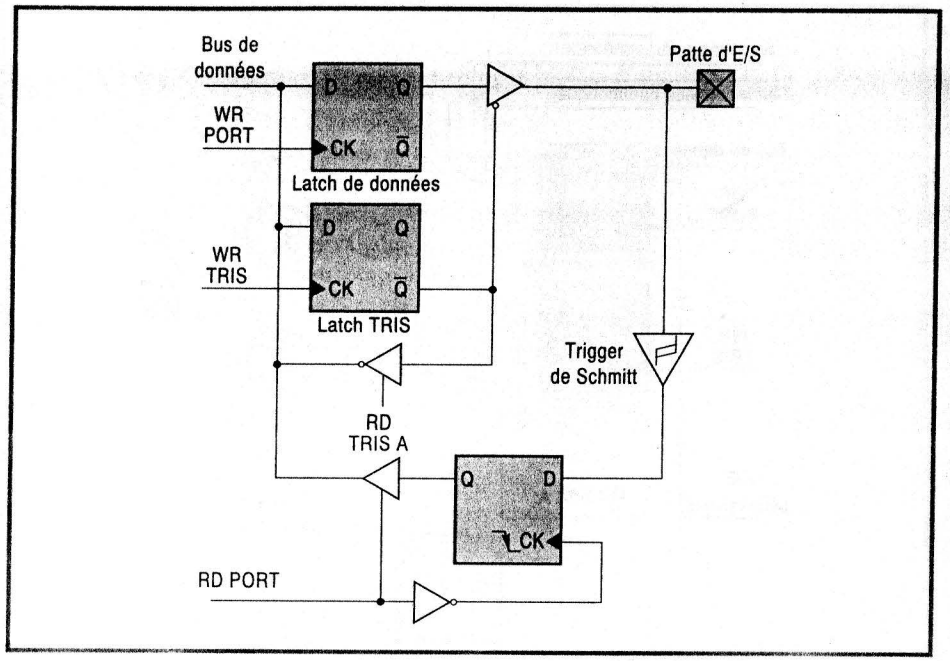


Figure 6.7 – Schéma équivalent des entrées/sorties du port D.

Ce port dispose d'un mode de fonctionnement particulier qui s'appelle le mode esclave parallèle (PSP pour *parallel slave port*) que nous décrirons ci-après.

Lorsqu'il ne fonctionne pas en mode esclave parallèle, le port D peut-être utilisé normalement. Le sens de travail de toutes ses lignes est alors contrôlé par le registre TRISD dans lequel un bit à 0 place la ligne correspondante en sortie et un bit à 1 la place en entrée. Suite à un *reset*, tous les bits du registre TRISD sont évidemment mis à 1.

6.6 Le port E

Le port E dispose de trois à huit lignes bidirectionnelles qui respectent le schéma interne visible *figure 6.8*. Elles sont munies d'un buffer TTL en sortie et d'un trigger de Schmitt en entrée.

Ces lignes peuvent être utilisées normalement comme toutes les lignes des ports parallèles ou servir de signaux de contrôle lorsque le port D est placé en mode

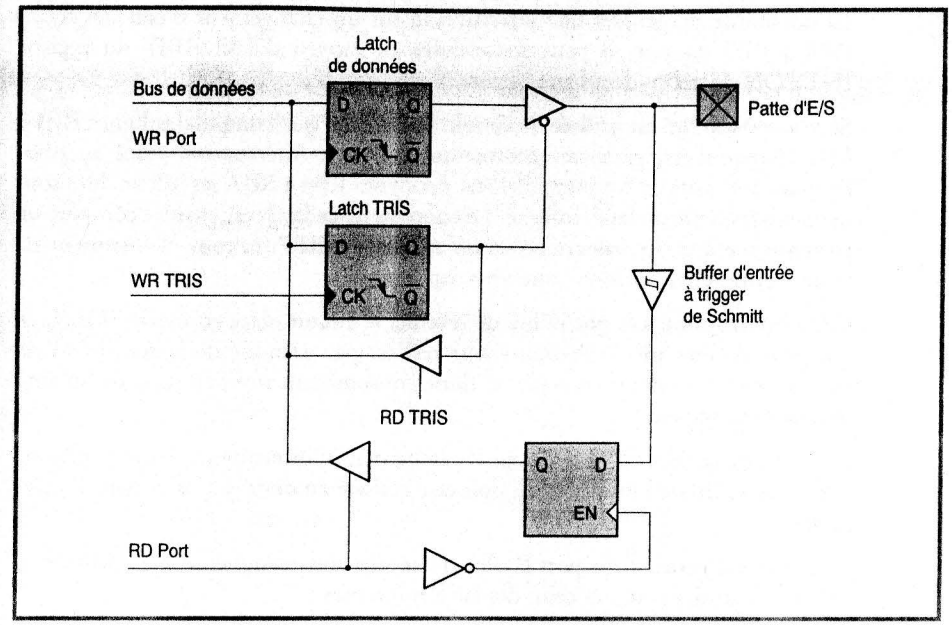


Figure 6.8 – Schéma équivalent des entrées/sorties du port E.

esclave parallèle (PSP). Dans ce cas, le registre TRISE ne sert plus à définir le sens de fonctionnement des lignes du port comme nous allons le voir ci-après.

Lorsqu'il ne fonctionne pas en mode esclave parallèle, le port E peut-être utilisé normalement. Le sens de travail de toutes ses lignes est alors contrôlé par le registre TRISE qui fonctionne normalement, à savoir qu'un bit à 0 place la ligne correspondante en sortie et qu'un bit à 1 la place en entrée. Suite à un *reset*, tous les bits du registre TRISE sont évidemment mis à 1.

6.7 Les ports F et G

Les ports F et G sont des ports unidirectionnels utilisables seulement en entrée. Ils partagent en effet leurs lignes avec la ressource interne de commande d'un afficheur à cristaux liquides qui utilise leurs lignes comme sorties de commande des segments de l'afficheur. La *figure 6.9*, valable pour le port F et le port G, rend bien compte de cette situation.

Comme vous pouvez le constater, seule la fonction entrée est présente pour les ports parallèles ; entrée réalisée ici encore au travers d'un trigger de Schmitt.

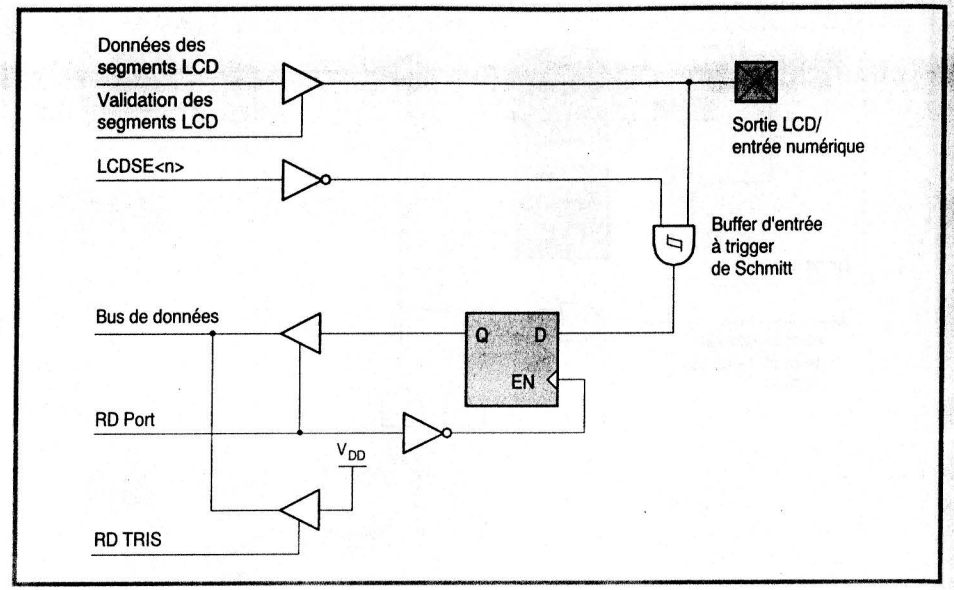


Figure 6.9 – Schéma équivalent des entrées/sorties des ports F et G ou ports LCD.

Tant qu'elles sont utilisées en entrées, ces lignes peuvent être exploitées normalement comme toutes les lignes des ports parallèles. Le fonctionnement en sortie est quant à lui sous le contrôle de l'interface pour afficheur à cristaux liquides et se trouve décrit au chapitre 13 lors de la présentation de cette ressource très particulière.

6.8 Le port GPIO

Ce port parallèle un peu particulier n'est présent, au moment où ces lignes sont écrites, que sur les PIC de la série 12Cxxx. Même s'il fonctionne dans son ensemble comme les autres ports parallèles, il dispose de fonctions supplémentaires imposées par le faible nombre de pattes du boîtier des PIC 12Cxxx et, donc, par le nécessaire partage de fonctions et de ressources qui en découle.

Il ne comporte que six lignes GP0 à GP5. Les bits 6 et 7 de son registre de données n'existent donc pas et sont lus comme étant à 0. Toutes les lignes peuvent être programmées en entrée ou en sortie individuellement de façon classique sous le contrôle du registre TRISGP, sauf GP3 qui est toujours une entrée. Le bit correspondant de TRISGP est donc toujours à 1 et ne peut être modifié.

Suite à un *reset*, et comme pour tous les autres ports parallèles, tous les bits de TRISGP sont évidemment initialisés à 1.

Comme le montre la ***figure 6.10***, le principe du port est identique à ce que nous avons déjà vu à maintes reprises, à savoir que les données de sortie sont « latchées » au niveau du registre de données alors que les données d'entrée sont lues directement sur la patte du port.

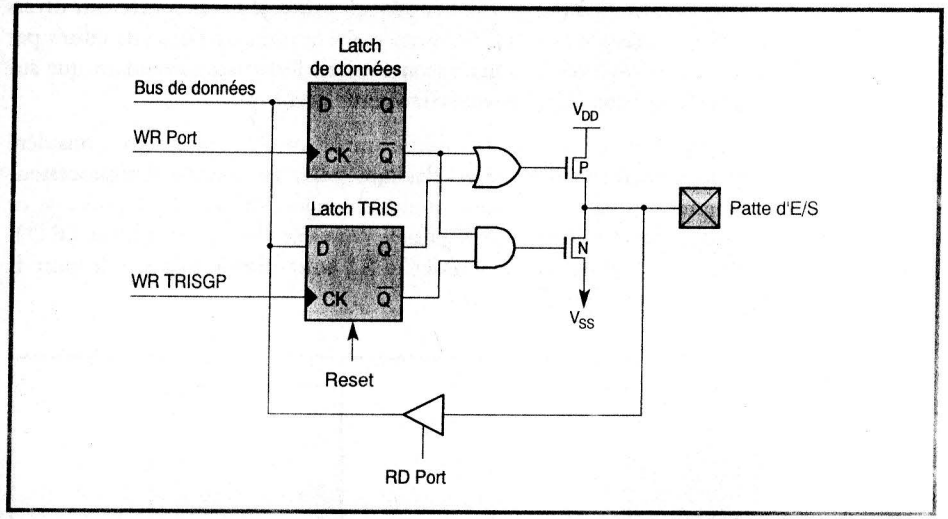


Figure 6.10 – Schéma équivalent des entrées/sorties de type GPIO.

Compte tenu des particularités des PIC 12Cxxx, le registre OPTION dont nous avons vu la description au chapitre 3 voit deux de ses bits changer de fonction en liaison avec l'existence de ce port parallèle GPIO :

- ***Bit 7 ou bit $\overline{GPWU}$ pour GPio enable wake up***

  Lorsqu'il est mis à 0, ce bit autorise le réveil du circuit depuis le mode SLEEP pour tout changement d'état des lignes GP0, GP1 ou GP3.

- ***Bit 6 ou bit $\overline{GPPU}$ pour GPio enable weak pull-up***

  Lorsqu'il est mis à 0 ce bit valide des résistances de tirage au niveau haut, de forte valeur, sur les pattes GP0, GP1 et GP3.

Les fonctions des autres bits du registre OPTION sont inchangées et conformes à ce que nous avons décrit au chapitre 3.

Enfin, si vous décidez de faire fonctionner le circuit en mode oscillateur externe, ce qui est choisi rappelons-le lors de la programmation du circuit, les pattes GP4

et GP5 sont automatiquement affectées à l'oscillateur d'horloge. Elles deviennent donc inutilisables par le port GPIO et les bits correspondants du registre TRISGP sont sans effet sur elles.

6.9 Le mode port parallèle esclave ou PSP

Ce mode de fonctionnement particulier peut en principe se rencontrer sur divers ports parallèles « classiques » des PIC à en croire certains documents édités par Microchip. Au moment où ces lignes sont écrites, il n'existe cependant que sur les ports D et E qui sont alors associés dans cette aventure.

Dans ce mode très intéressant, le microcontrôleur PIC peut être considéré comme étant un circuit d'interface classique pour un autre microprocesseur 8 bits. Les lignes du port D deviennent alors les lignes du bus de données de ce microprocesseur tandis que les trois lignes de contrôle classiques : lecture ($\overline{RD}$), écriture ($\overline{WR}$) et validation du boîtier ($\overline{CS}$) sont disponibles sur le port E comme cela est schématisé *figure 6.11.*

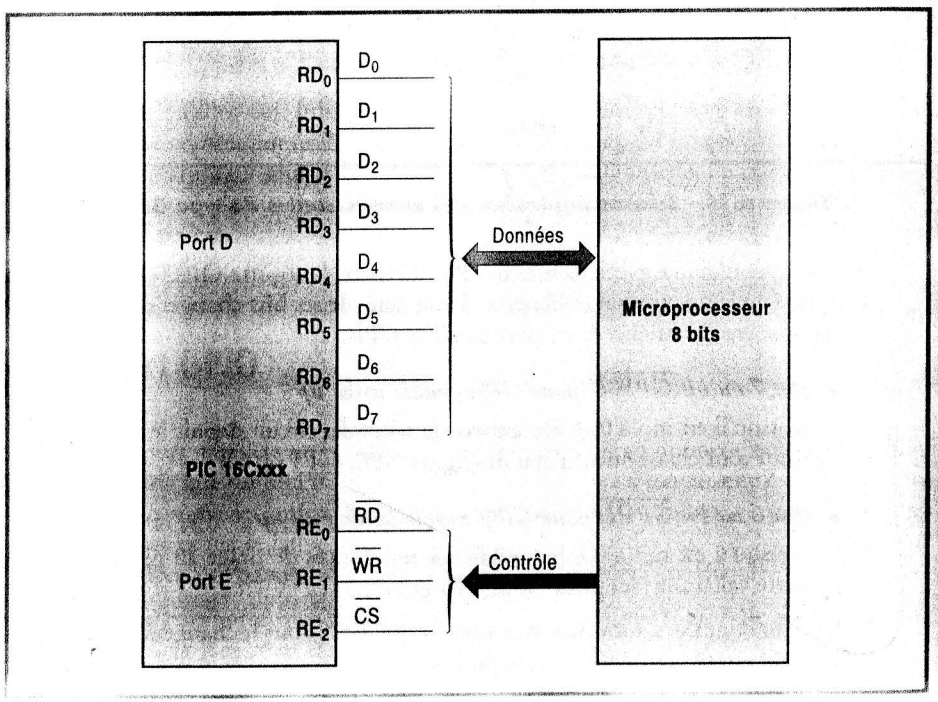


Figure 6.11 – Principe de connexion d'un PIC en mode esclave parallèle.

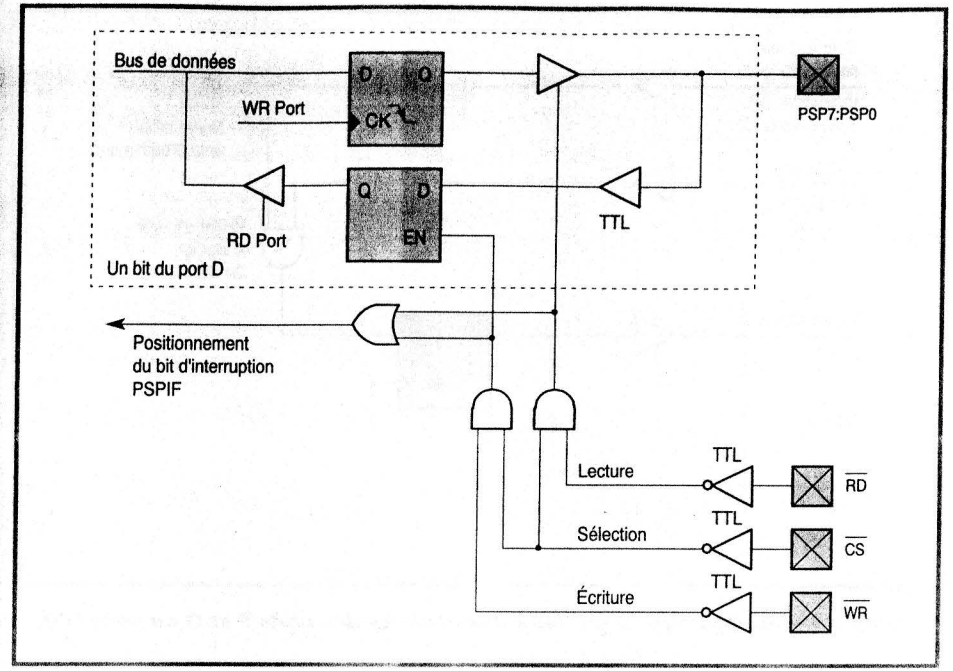


Figure 6.12 – Schéma équivalent des ports D et E en mode esclave parallèle.

Le schéma équivalent à la combinaison des ports D et E utilisés dans ce mode est visible quant à lui *figure 6.12.*

Lorsqu'ils sont utilisés dans ce mode, les circuits d'entrée des ports D et E comportent des « buffers » TTL. Par ailleurs, et contrairement à ce qui se passe lors de l'utilisation des ports parallèles classiques, on dispose ici d'un latch de données en sortie et d'un latch de données en entrée. C'est dans ces derniers que sont lues et écrites les données qui transitent sur le « bus de données » que constitue alors le port D.

Compte tenu du fait que le sens de transfert des données sur le port D est sous contrôle des lignes externes $\overline{RD}$ et $\overline{WR}$, le registre TRISD est ignoré et son contenu est sans effet sur le port D.

Par ailleurs, le fonctionnement de ce port parallèle esclave ou PSP est contrôlé par le registre TRISE, dont les bits adoptent pour la circonstance des fonctions particulières que voici.

Accès E : Écriture L : Lecture	L	L	L/E	L/E	L	L/E	L/E	L/E
TRISE	IBF	OBF	IBOV	PSPMODE	—	TRISE2	TRISE1	TRISE0
État à la mise sous tension	0	0	0	0	0	1	1	1

- ***Bits 0 à 2 ou bits TRISE0 à TRISE2***

  Ils contrôlent le sens de travail des lignes RE0 à RE2 de manière classique : un 0 place la ligne correspondante en sortie et un 1 la place en entrée.

- ***Bit 3***

  Non utilisé et lu comme étant à 0.

- ***Bit 4 ou bit PSPMODE pour parallel slave port MODE***

  S'il est mis à 1, ce bit place le port D en mode esclave parallèle. Un 0 laisse le port D fonctionner normalement.

- ***Bit 5 ou bit IBOV pour input buffer overflow***

  Ce bit n'a de signification qu'en mode esclave parallèle. Il passe à 1 si le circuit externe a écrit une donnée sur le port D alors que la précédente n'avait pas encore été lue par le PIC. Ce bit doit ensuite être remis à 0 par logiciel.

- ***Bit 6 ou bit OBF pour output buffer full***

  Ce bit n'a de signification qu'en mode esclave parallèle. Il reste à 1 tant que le buffer de sortie contient toujours la donnée qui y a été placée par le PIC, c'est-à-dire tant que le circuit externe n'a pas lu le port D.

- ***Bit 7 ou bit IBF pour input buffer full***

  Ce bit n'a de signification qu'en mode esclave parallèle. Il passe à 1 lorsque le buffer d'entrée contient une donnée valide qui n'a pas encore été lue par le PIC, c'est-à-dire lorsque le circuit externe a écrit dans le port D.

Notez que lors du fonctionnement en mode esclave parallèle, les lignes du port E adoptent la configuration suivante :

- RE0 est la ligne $\overline{RD}$, c'est-à-dire qu'un niveau bas doit lui être appliqué pour lire le port D ;
- RE1 est la ligne $\overline{WR}$, c'est-à-dire qu'un niveau bas doit lui être appliqué pour écrire dans le port D ;
- RE2 est la ligne $\overline{CS}$, c'est-à-dire qu'un niveau bas doit lui être appliqué pour valider les lignes de données du port D.

Ces affectations et fonctions correspondent bien à ce que l'on a l'habitude de trouver sur n'importe quel circuit d'interface pour microprocesseur 8 bits.

Pour pouvoir fonctionner en mode esclave parallèle, et hormis la mise à 1 du bit PSPMODE, il faut également placer les lignes RE0 à RE2 en entrées par écriture d'un 1 dans les bits 0 à 2 du registre TRISE.

Notez également que, comme nous l'avons vu lors de la description des registres PIE au chapitre 3, il est possible de faire générer une interruption lors de toute lecture ou écriture sur le port D configuré dans ce mode, interruption signalée par le bit PSPIF du registre PIRx correspondant.

Les *figures 6.13* et *6.14* confirment nos explications en vous proposant les chronogrammes d'une écriture et d'une lecture (vues de l'extérieur) sur le port D utilisé en mode PSP. Ces chronogrammes mettent bien en évidence le rôle des signaux externes sur les bits d'état du registre TRISE.

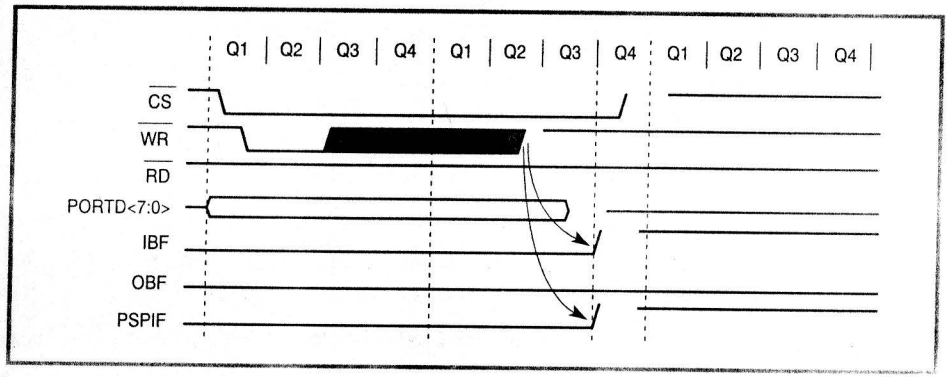


Figure 6.13 – Chronogramme d'écriture dans un PIC en mode esclave parallèle.

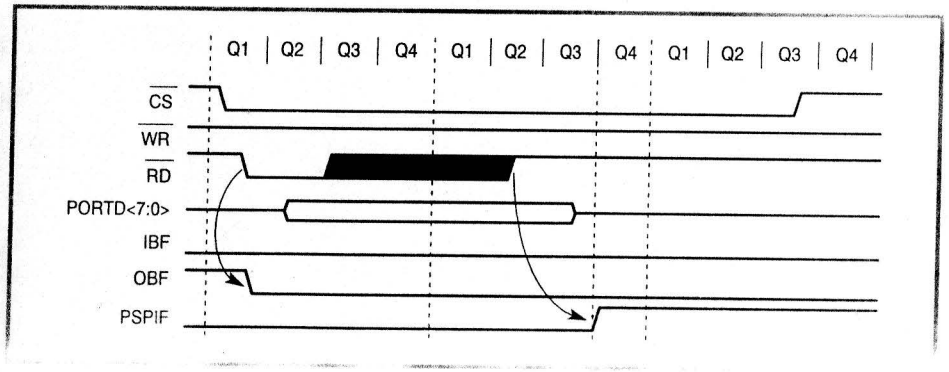


Figure 6.14 – Chronogramme de lecture dans un PIC en mode esclave parallèle.

CHAPITRE 7

LES *TIMERS*

Selon la version choisie, un microcontrôleur PIC peut contenir de un à trois *timers* appelés *timer 0, timer 1* et *timer 2*. Le *timer 0* est toujours présent, même sur les circuits les plus simples et, en raison d'une survivance du passé due aux PIC 16C5x, il s'appelle encore RTCC dans certaines documentations, ce qui est l'abréviation américaine de son appellation première qui était « horloge temps réel ».

Chaque *timer* peut générer une interruption comme nous l'avons vu lors de la présentation des registres INTCON et PIEx au chapitre 3. Si le *timer 0* reste relativement simple, les *timers 1* et *2* peuvent par contre être associés à des modules fonctionnels appelés CCP1, CCP2 et PWM, présentés au chapitre 8, qui permettent de réaliser de la capture de données en entrée, de la comparaison de données en sortie et de la modulation de largeur d'impulsions. Toutes ces fonctionnalités sont donc comparables à celles que l'on rencontre traditionnellement sur les *timers* les plus performants des microcontrôleurs 8 bits actuels.

Ces trois *timers* fonctionnant de ce fait de façon très différente, nous allons les étudier indépendamment l'un de l'autre et tour à tour, en commençant bien entendu par celui qui est le plus fréquemment rencontré, à savoir le *timer 0*.

7.1 Le *timer 0*

Le *timer 0* est le plus simple des trois car, comme nous l'avons dit, il est en fait quasiment identique à la RTCC présente sur les « vieux » PIC 16C5x. La principale nouveauté le concernant est la possibilité de générer une interruption qui se produit lorsque le *timer* déborde de FF à 00. Le mode de fonctionnement de ce *timer* est relativement simple, dès lors que l'on prend connaissance de la *figure 7.1* qui présente son synoptique interne.

Si l'on part du registre TMR0 qui est, comme nous l'avons vu au chapitre 3, le registre d'adresse 01, on constate qu'il peut recevoir son signal directement ou

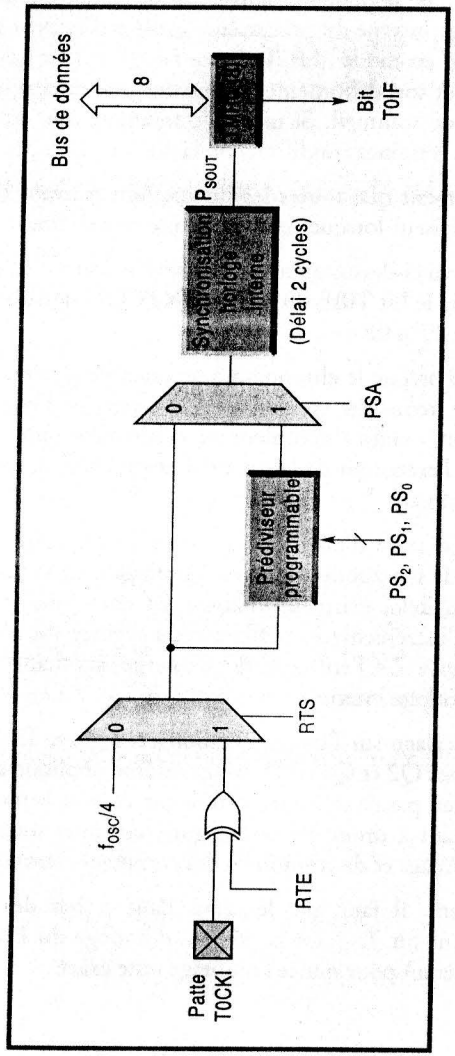


Figure 7.1 – Schéma équivalent du timer 0.

après passage par un prédiviseur, la sélection étant faite par un bit appelé PSA contenu dans le registre OPTION.

Si le prédiviseur est sélectionné, son taux de division est programmé au moyen de trois bits ayant pour noms PS0, PS1 et PS2, également contenus dans le registre OPTION.

Dans tous les cas, le signal utilisé peut à nouveau provenir de deux sources selon l'état du bit de sélection T0CS. Soit il provient de l'oscillateur d'horloge après division de fréquence par quatre, il est donc bien à la fréquence de l'horloge système ou horloge instruction, soit il provient de l'extérieur via la patte T0CKI. Le OU EXCLUSIF commandé par le bit T0SE permet quant à lui de choisir si l'incrémentation du registre TMR0 aura lieu sur un front montant ou descendant du signal appliqué à T0CKI.

Les bits T0SE, T0CS, PSA et PS0 à PS2 qui servent en fait à configurer ce *timer 0* sont contenus dans le registre OPTION dont voici rappelée la signification :

Accès E : Écriture L : Lecture	L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E
OPTION	RBPU	INTDEG	T0CS	T0SE	PSA	PS2	PS1	PS0
État à la mise sous tension	1	1	1	1	1	1	1	1

- ***Bits 0 à 2 ou bits PS0, PS1 et PS2 pour prescaler***

  Ces trois bits servent à définir le taux de prédivision appliqué à l'entrée du *timer 0* ou du *timer* chien de garde, conformément aux indications du *tableau 7.1.* Attention ! Ce taux est différent selon que le prédiviseur est affecté à l'un ou l'autre. Par ailleurs, notez que pour bénéficier d'un taux de prédivision de 1 sur le *timer 0,* il faut affecter le prédiviseur au *timer* chien de garde au moyen du bit PSA.

- ***Bit 3 ou bit PSA pour prescaler assignment***

  Ce bit permet d'affecter le prédiviseur au *timer 0* lorsqu'il est mis à 0 ou au *timer* chien de garde lorsqu'il est mis à 1.

- ***Bit 4 ou bit T0SE pour timer 0 source edge***

  Lorsqu'il est à 0, ce bit indique que le *timer 0* est incrémenté sur un front descendant de son signal d'entrée. Il est incrémenté sur un front montant lorsque ce bit est mis à 1.

- ***Bit 5 ou bit T0CS pour timer 0 clock select***

  Lorsqu'il est à 0, ce bit sélectionne l'horloge instruction CLKOUT comme horloge du *timer 0*. Il sélectionne l'entrée T0CKI lorsqu'il est mis à 1.

Les autres bits de ce registre ne concernent pas le *timer 0* et ont été vus au chapitre 3.

Tableau 7.1 – Taux de division du prédiviseur du *timer 0* ou du *timer* chien de garde en fonction des bits PS0 à PS2 du registre OPTION.

PS2	PS1	PS0	*Timer 0*	Chien de garde
0	0	0	2	1
0	0	1	4	2
0	1	0	8	4
0	1	1	16	8
1	0	0	32	16
1	0	1	64	32
1	1	0	128	64
1	1	1	256	128

Le registre TMR0 augmente évidemment d'une unité à chaque coup de l'horloge sélectionnée par le registre OPTION, aux quelques restrictions près que nous allons détailler ci-dessous. Lorsqu'il arrive à FF, il revient tout simplement à 00 en générant une interruption si elle est autorisée, et il continue son cycle indéfiniment.

Le registre TMR0 peut évidemment être lu ou écrit directement avec n'importe quelle instruction afin de connaître sa position courante ou pour le faire partir d'un état déterminé. Il importe de savoir que, après toute écriture dans ce registre, un délai de trois cycles d'instructions est nécessaire pour que l'incrémentation reprenne. Ce délai est indépendant de la source d'horloge sélectionnée.

Les instructions concernées sont les seules instructions d'écriture et de lecture-modification-écriture telle que MOVF TMR0 ou CLRF TMR0. Pour tester le passage à zéro de ce registre sans affecter le déroulement régulier du comptage, il est donc conseillé d'utiliser un MOVF TMR0,W par exemple, qui n'effectue qu'une lecture.

© Dunod. La photocopie non autorisée est un délit

La *figure 7.2* montre ainsi le chronogramme de fonctionnement du *timer 0* en mode horloge interne et sans prédivision afin de mettre en évidence ce temps de latence.

Compte tenu de la présence dans le *timer 0* d'une circuiterie de synchronisation avec l'horloge interne du processeur, le *timer 0* ne peut plus fonctionner lorsque ce dernier est en mode SLEEP. De ce fait, il ne faut pas compter sur l'évolution de ce *timer* et son débordement générant une interruption pour faire sortir un PIC du mode sommeil. Si une telle fonction vous intéresse, il faut utiliser le *timer 1* sous certaines conditions précisées lors de sa présentation.

Notez également que toutes les instructions écrivant dans TMR0 remettent à zéro le prédiviseur lorsque celui-ci est affecté à ce *timer*.

Nous avons vu ci-dessus qu'une interruption pouvait être générée, sous réserve de validation via le bit T0IE du registre INTCON, lorsque le compteur du *timer 0* débordait de FF à 00.

La *figure 7.3* précise le chronogramme exact de génération de cette interruption car, compte tenu des mécanismes internes de l'unité centrale, deux cycles d'instructions « vides » s'écoulent nécessairement entre le débordement proprement dit et l'exécution de la première instruction du programme de traitement des interruptions.

Lors de l'utilisation d'une horloge externe, via la patte T0CKI, et en raison du mécanisme de synchronisation avec l'horloge interne qui se produit au niveau de ce *timer*, un délai existe inévitablement entre une transition valide de cette horloge et l'incrémentation effective du registre du *timer*. En effet, comme le montre la *figure 7.4*, l'entrée horloge externe est échantillonnée lors des cycles Q_2 et Q_4 de l'horloge interne.

Outre le décalage sur l'incrémentation du registre TRM0, cet échantillonnage lors des phases Q2 et Q4 de l'horloge interne implique que le signal appliqué sur T0CKI ne soit pas de durée trop faible car, comme le montre la courte impulsion visible en haut à droite de cette figure, certaines transitions risquent alors de passer inaperçues et de conduire à des comptages erronés.

Concrètement, il faut que les états haut et bas de T0CKI durent plus de $2T_{OSC} + 20$ ns où T_{OSC} est la période d'horloge du PIC (horloge réelle et non horloge système) pour que le comptage reste exact.

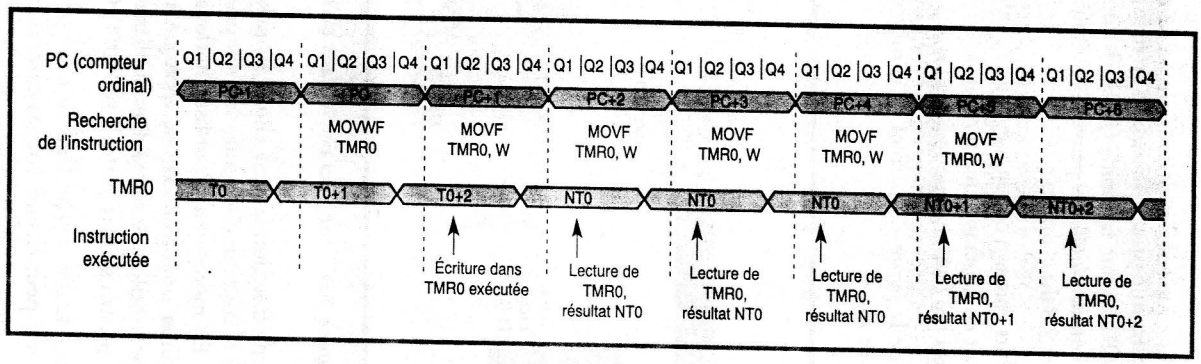


Figure 7.2 - Chronogramme de fonctionnement du timer 0 en mode horloge interne.

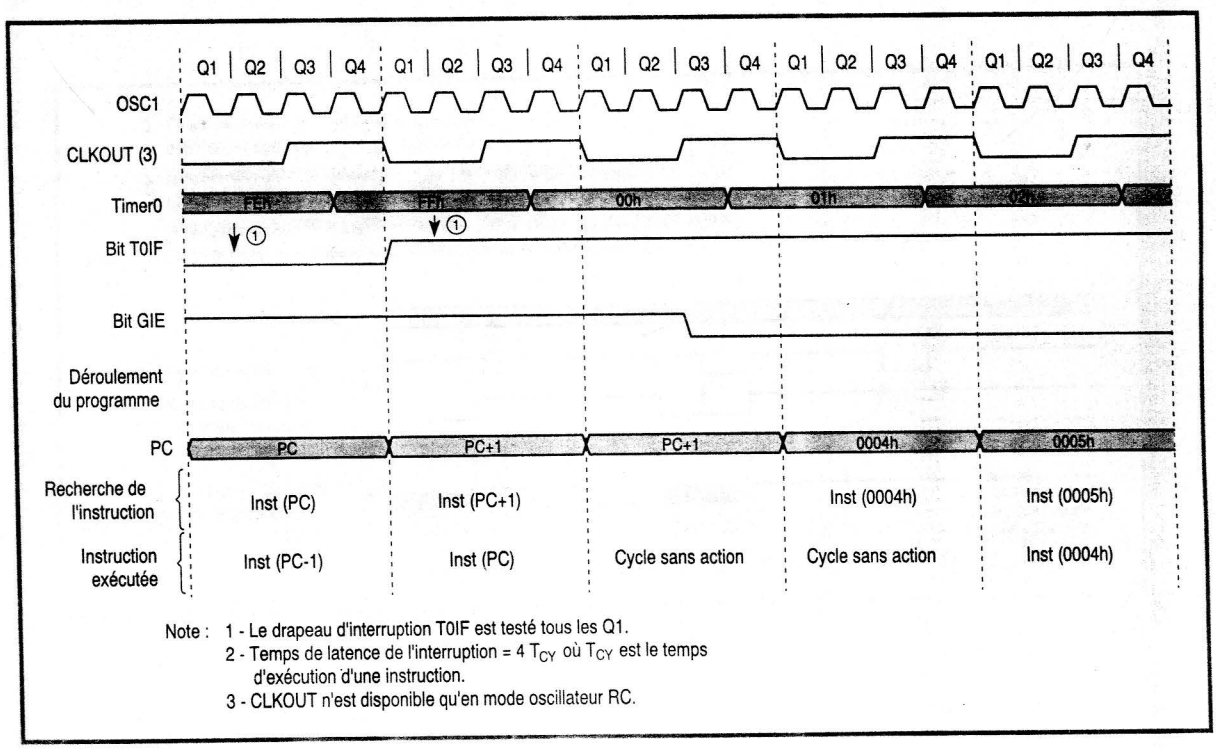


Figure 7.3 - Chronogramme de génération d'interruption à partir du timer 0.

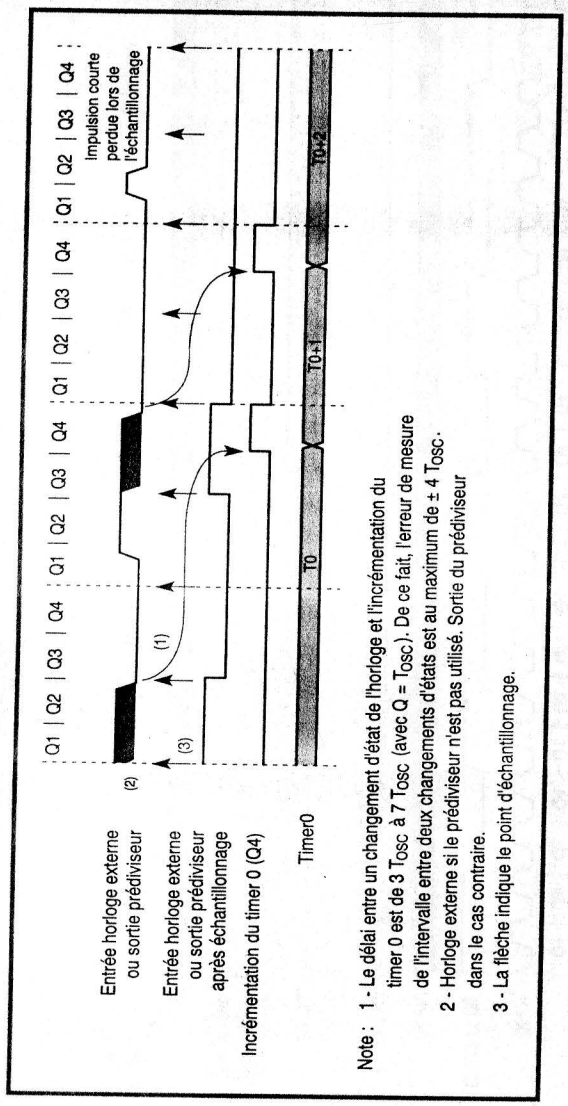


***Figure 7.4 – Chronogramme d'échantillonnage de l'horloge externe du* timer 0.**

7.2 Le *timer 1*

Le *timer 1* est constitué principalement d'un registre 16 bits divisé en deux registres 8 bits : TMR1L pour les poids faibles (L comme *low*) et TMR1H pour les poids forts (H comme *high*). Ils peuvent être tous deux lus et écrits aux adresses présentées *figure 3.8* du chapitre 3.

Ce registre 16 bits voit son contenu évoluer de 0000 à FFFF, puis revenir à 0000 par débordement en générant une interruption à ce moment-là, s'il y a été autorisé au moyen du bit TMR1IE du registre PIEx.

Le synoptique de la *figure 7.5* permet de comprendre les divers modes de fonctionnement de ce *timer*.

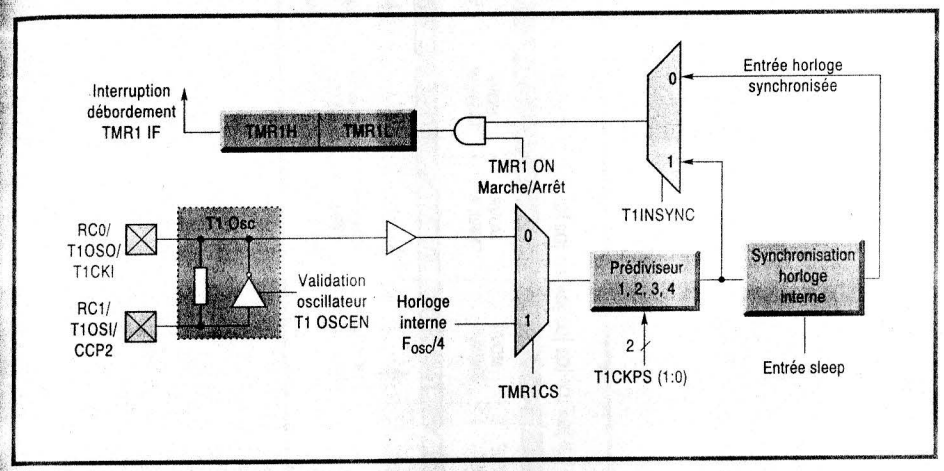


***Figure 7.5 – Schéma équivalent du* timer 1.**

Selon le mode de fonctionnement choisi au moyen du registre T1CON dont nous allons voir le contenu ci-dessous, le *timer* peut utiliser trois sources d'horloges différentes. En mode *timer*, il utilise l'horloge instruction. En mode compteur, il peut utiliser un signal externe appliqué à la patte T1OSO/T1CKI, ou bien encore il peut utiliser le signal d'un oscillateur qui lui est propre, réalisé en connectant un quartz entre les pattes T1OSO/T1CKI et T1OSI.

Dans tous les cas, le signal d'horloge du *timer* passe par un prédiviseur de taux programmable et il peut ensuite être synchronisé ou non avec l'horloge interne. Cette synchronisation est évidemment parfaitement inutile lorsque l'on utilise déjà cette source comme horloge. Par contre, en mode horloge externe (ou horloge délivrée par son propre oscillateur), vous êtes libre de choisir ou non cette synchronisation.

© Dunod. La photocopie non autorisée est un délit.

Les différents modes de fonctionnement de ce *timer* sont contrôlés par le registre T1CON dont chaque bit a la signification particulière que voici :

Accès E : Écriture L : Lecture	L	L	L/E	L/E	L/E	L/E	L/E	L/E
T1CON	—	—	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON
État à la mise sous tension	0	0	0	0	0	0	0	0

- ***Bit 0 ou bit TMR1ON pour timer 1 ON***

  Si ce bit est mis à 1, le *timer 1* fonctionne. Il est arrêté dans le cas contraire.

- ***Bit 1 ou bit TMR1CS pour timer 1 clock select***

  Ce bit sélectionne l'horloge externe appliquée à T1OSO/T1CKI (front montant) s'il est mis à 1, et l'horloge interne dans le cas contraire.

- ***Bit 2 ou bit T1INSYNC pour timer 1 input synchronization***

  S'il est mis à 1, ce bit interdit la synchronisation d'horloge. Il l'autorise dans le cas contraire. Ce bit est sans effet en mode horloge interne, c'est-à-dire lorsque TMR1CS est à 0.

- ***Bit 3 ou bit T1OSCEN pour timer 1 oscillator enable***

  S'il est mis à 1, ce bit valide l'oscillateur réalisé entre les pattes T1OSCO/T1CKI et T1OSI. Dans le cas contraire, l'amplificateur connecté entre ces deux pattes est désactivé pour réaliser des économies de consommation.

- ***Bits 4 et 5 ou bits T1CKPS0 et T1CKPS1 pour timer 1 clock prescaler***

  Ces bits définissent le taux de fonctionnement du prédiviseur selon les indications du *tableau 7.2.*

- ***Bits 6 et 7***

  Non utilisés et lus comme étant à zéro.

Lorsqu'il est utilisé en mode *timer*, le *timer 1* utilise comme horloge l'horloge système, aucune synchronisation n'est donc nécessaire puisque celle-ci est tout naturellement réalisée.

Lorsqu'il est utilisé en mode compteur, le registre TMR1 augmente d'une unité à chaque front montant du signal appliqué à T1CKI ou du signal produit par l'oscillateur « local » selon celle de ces options qui a été choisie.

Tableau 7.2 – Taux de division du prédiviseur du *timer 1* en fonction des bits T1CKPS0 et T1CKPS1 du registre T1CON.

T1CKPS1	T1CKPS0	Taux de division
0	0	1
0	1	2
1	0	4
1	1	8

Si la synchronisation est validée, il est possible d'utiliser le *timer 1* conjointement au module CCP présenté au chapitre suivant. Par contre, le *timer 1* s'arrête de fonctionner en mode SLEEP, puisque la synchronisation ne peut plus avoir lieu du fait de l'arrêt de l'horloge interne dans ce mode.

Si la synchronisation n'est pas validée, le fonctionnement avec les modules CCP n'est pas possible ; par contre le *timer* continue à compter même en mode SLEEP et peut donc « réveiller » le microcontrôleur et le faire automatiquement sortir de ce mode lors de son débordement si vous avez autorisé la génération d'une interruption dans cette situation.

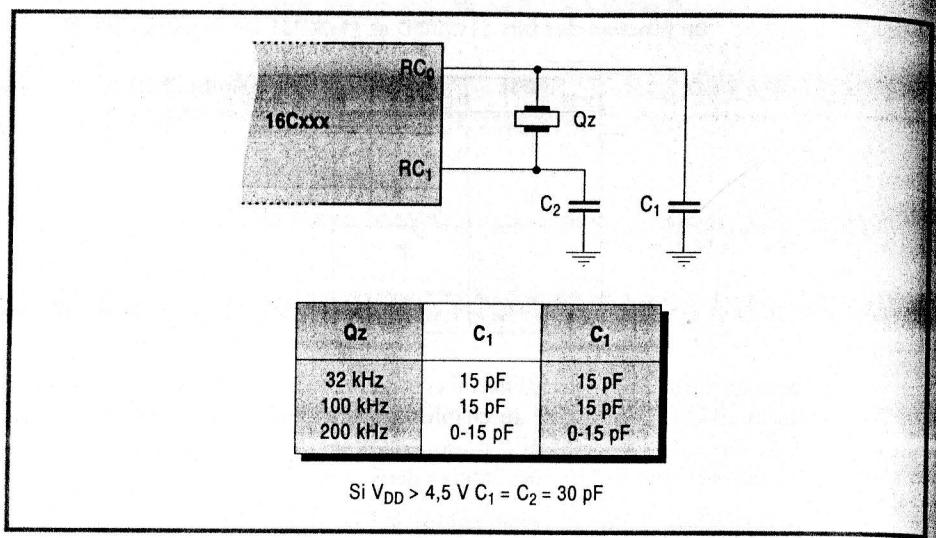
La synchronisation fonctionne selon le même principe que celui vu ci-dessus pour le *timer 0.* Il faut donc que les états haut et bas du signal appliqué sur T1CKI, ou produits par l'oscillateur « local », durent plus de $2T_{OSC} + 20$ ns où T_{OSC} est la période d'horloge du PIC (horloge réelle et non horloge système) pour que le comptage reste exact.

L'oscillateur local qui peut être réalisé par connexion d'un quartz entre les pattes T1OSO/T1CKI et T1OSI doit respecter le schéma présenté *figure 7.6.* La fréquence maximum de fonctionnement prévue actuellement est de 200 kHz et ce circuit est optimisé pour un quartz à 32 kHz. Cette même figure précise les valeurs des composants passifs à utiliser pour trois fréquences typiques.

7.3 Le *timer 2*

Comme le montre le synoptique de la *figure 7.7*, ce *timer* est composé d'un registre 8 bits appelé TMR2, associé à un prédiviseur et à un postdiviseur ainsi qu'à un registre dit de période et appelé PR2.

Le registre TMR2 voit son contenu partir de 00 et être augmenté au rythme de l'horloge instruction, c'est-à-dire de l'horloge du PIC divisée par quatre, après



Qz	C_1	C_1
32 kHz	15 pF	15 pF
100 kHz	15 pF	15 pF
200 kHz	0-15 pF	0-15 pF

Si $V_{DD} > 4,5$ V $C_1 = C_2 = 30$ pF

Figure 7.6 – Schéma d'utilisation de l'oscillateur local du timer 1.

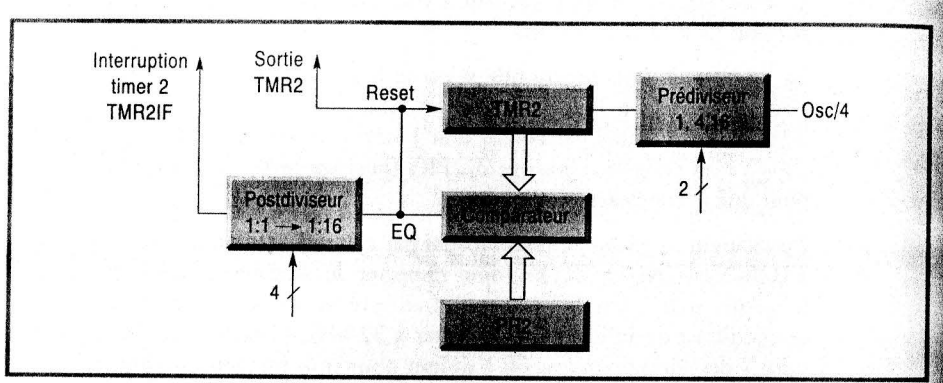


Figure 7.7 – Schéma équivalent du timer 2.

division éventuelle par le prédiviseur. Lorsque le contenu de TMR2 devient égal au contenu du registre période PR2, il est remis à zéro et un signal est appliqué au postdiviseur. La sortie de ce dernier peut alors générer une interruption si elle y a été autorisée par le bit TMRIE du registre PIEx adéquat comme nous l'avons vu au chapitre 3.

Notez également que la sortie de ce postdiviseur peut être utilisée par une autre ressource interne, en l'occurrence le port série synchrone ou SSP, afin de servir d'horloge de transmission.

© Dunod. La photocopie non autorisée est un délit.

Les contenus des prédiviseur et postdiviseur sont automatiquement mis à zéro lors de toute écriture dans le registre TMR2 ou dans le registre de contrôle T2CON, ainsi que lors d'un *reset*, quelle qu'en soit l'origine.

Le fonctionnement du *timer 2* est contrôlé au moyen du registre T2CON dont chaque bit a la signification particulière que voici :

Accès E : Écriture L : Lecture	L	L/E	L/E	L/E	L/E	L/E	L/E	L/E
T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0
État à la mise sous tension	0	0	0	0	0	0	0	0

- *Bits 0 et 1 ou bits T2CKPS0 et T2CKPS1 pour timer 2 clock prescaler*

  Ils fixent le taux de division appliqué au signal d'horloge selon les indications du *tableau 7.3.*

- *Bit 2 ou bit TMR2ON pour timer 2 ON*

  Ce bit autorise le fonctionnement du *timer 2* lorsqu'il est mis à 1 et l'arrête dans le cas contraire.

- *Bits 3 à 6 ou bits TOUTPS0 à TOUTPS3 pour timer output postscaler*

  Ces bits sélectionnent le taux de division du postdiviseur selon les indications du *tableau 7.4.*

Tableau 7.4 – Taux de division du prédiviseur du *timer 2* en fonction des bits T2CKPS0 et T2CKPS1 du registre T2CON.

T2CKPS2	T2CKPS1	Taux de division
0	0	1
0	1	4
1	X	16

- *Bit 7*

  Non utilisé et lu comme étant à 0.

En plus de leur utilisation en mode autonome, les *timers 1* et *2* peuvent être associés aux modules de comparaison, de capture et de modulation de largeur d'impulsions que nous allons vous présenter au chapitre suivant.

Tableau 7.3 – Taux de division du postdiviseur du *timer 2* en fonction des bits TOUTPS3 à TOUTPS0 du registre T2CON.

TOUTPS3	TOUTPS2	TOUTPS1	TOUPS0	Taux de division
0	0	0	0	1
0	0	0	1	2
0	0	1	0	3
0	0	1	1	4
0	1	0	0	5
0	1	0	1	6
0	1	1	0	7
0	1	1	1	8
1	0	0	0	9
1	0	0	1	10
1	0	1	0	11
1	0	1	1	12
1	1	0	0	13
1	1	0	1	14
1	1	1	0	15
1	1	1	1	16

© Dunod. La photocopie non autorisée est un délit.

CHAPITRE 8

LES MODULES DE CAPTURE ET COMPARAISON (CCP) ET DE MODULATION DE LARGEUR D'IMPULSIONS (PWM)

Un certain nombre de circuits disposent d'un module capture, comparaison et modulation de largeur d'impulsions qui, en utilisant une partie des ressources des *timers 1* et *2*, permet de réaliser avec une grande souplesse de la mesure de temps, de période ou de fréquence et de la génération de signaux périodiques calibrés. L'intérêt de ce module est de fonctionner de manière quasiment automatique, libérant ainsi au maximum l'unité centrale pour d'autres tâches.

Ce module s'appelle CCP dans les documentations Microchip, pour *Capture, Compare* PWM, et il peut être présent en un ou plusieurs exemplaires identiques selon les versions de circuits PIC. Dans tous les cas, il utilise trois registres et ceux-ci sont donc numérotés en fonction du nombre de modules CCP présents dans le circuit comme indiqué *tableau 8.1*.

Tableau 8.1 – Affectation des registres aux divers modules de capture et de comparaison.

Nom générique	CCP1	CCP2	Commentaire
CCPxCON	CCP1CON	CCP2CON	Registre de contrôle
CCPRxH	CCPR1H	CCPR2H	Octet haut registre CCP
CCPRxL	CCPR1L	CCPR2L	Octet bas registre CCP
CCPx	CCP1	CCP2	Patte CCP

Le registre CCPxCON est un registre de contrôle définissant le mode de fonctionnement de chaque module, tandis que les registres CCPRxL et CCPRxH forment en fait un registre CCPRx sur 16 bits qui est un registre de comptage.

Ce module CCP utilise une partie des ressources des *timers 1* et *2* selon la fonction réalisée conformément aux indications du *tableau 8.2*. Il est donc évident que le *timer* correspondant ne pourra pas être utilisé librement en même temps que la fonction du module CCP correspondante, et il importe donc d'en tenir compte.

Tableau 8.2 – Exploitation des *timers 1* et *2* selon la fonction choisie.

Mode CCP	Ressource utilisée
Capture	*Timer 1*
Comparaison	*Timer 1*
PWM	*Timer 2*

Enfin, lorsque deux modules CCP distincts sont présents dans un même circuit, il n'est pas toujours possible de choisir librement la fonction de chacun d'eux, du fait justement de l'utilisation des ressources des *timers 1* et *2*. Le *tableau 8.3* montre ainsi l'interaction qui existe entre deux modules CCP distincts selon les fonctions exécutées par chacun d'eux.

Tableau 8.3 – Interactions possibles entre deux modules CCP.

Mode CCPx	Mode CCPy	Interaction
Capture	Capture	Même base de temps *timer 1*
Capture	Comparaison	La comparaison doit être programmée en mode déclenchement spécial qui remet TMR1 à 0.
Comparaison	Capture	La comparaison doit être programmée en mode déclenchement spécial qui remet TMR1 à 0.
PWM	PWM	La fréquence est la même.
PWM	Capture	Aucune
PWM	Comparaison	Aucune

Ces limitations sont toutefois assez peu contraignantes lorsque l'on connaît la puissance des différents modes de travail des modules CCP que nous allons découvrir maintenant.

© Dunod. La photocopie non autorisée est un délit.

8.1 Le mode capture

Dans ce mode, le registre CCPRx capture le contenu du *timer 1* à l'une des échéances suivantes sur la patte CCPx correspondante :

- tous les fronts descendants ;
- tous les fronts montants ;
- tous les quatre fronts montants ;
- tous les seize fronts montants.

Le choix entre ces divers événements est réalisé grâce aux bits contenus dans le registre CCPxCON que nous verrons dans un instant.

Lorsqu'une capture a lieu, une interruption peut être générée sous réserve d'y être autorisée par le bit approprié du registre PIEx correspondant présenté au chapitre 3. Si une autre capture a lieu avant que le contenu de CCPRx n'ait été lu, ce contenu est écrasé par le nouveau. Par contre, une capture ne remet pas à 0 le contenu des registres du *timer 1* qui continue à évoluer normalement. Cela permet, sous certaines conditions, de l'utiliser simultanément à d'autres tâches.

Pour que le système fonctionne correctement, le *timer 1* doit fonctionner en mode *timer* ou en mode compteur synchronisé par l'horloge interne. S'il fonctionne en mode non synchronisé, la capture peut ne pas se dérouler correctement.

Notez également que pour que ce système fonctionne correctement, la patte CCPx doit évidemment être programmée en entrée. Si elle est configurée en sortie et qu'une écriture dans le registre du port parallèle avec lequel elle est commune a lieu, une fausse capture peut être générée.

La *figure 8.1* montre de façon très visuelle le fonctionnement de ce mode de capture.

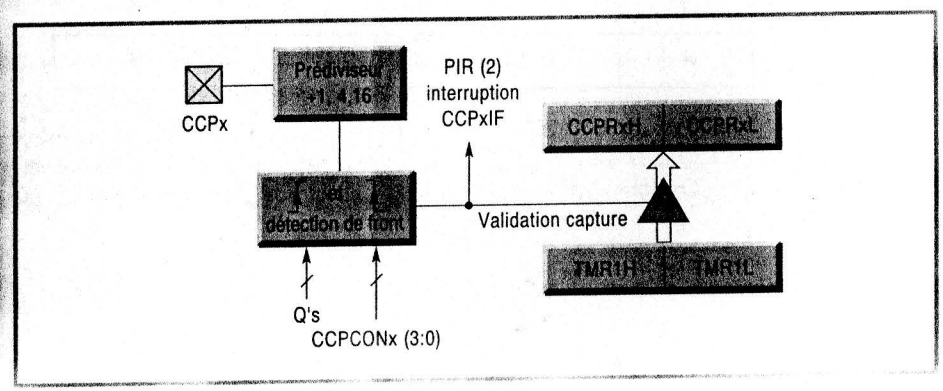


Figure 8.1 – Schéma équivalent du module de capture.

Remarquez la présence du prédiviseur qui permet la capture tous les fronts, tous les quatre fronts ou tous les seize fronts. Son contenu est mis à zéro lors d'un *reset* ainsi que lors de tout arrêt du module CCP.

Prenez garde au fait que, si vous changez la valeur de ce prédiviseur afin de définir d'autres conditions de capture, une fausse interruption de capture peut être générée. Il est donc conseillé d'arrêter le module CCP avant de changer les conditions de capture pour éviter ce phénomène.

8.2 Le mode comparaison

Dans ce mode, le contenu du registre CCPRx est comparé en permanence avec le contenu du *timer 1*. Lorsque l'égalité a lieu, les phénomènes suivants peuvent se produire :

- passage au niveau haut de la patte CCPx correspondante ;
- passage au niveau bas de la patte CCPx correspondante ;
- aucun changement d'état de la patte, mais génération d'une interruption si bien sûr le module CCP y est autorisé.

Dans les deux premiers cas, la patte correspondante du port parallèle avec lequel elle est commune doit évidemment être programmée en sortie par mise à zéro du bit correspondant du registre TRISx.

La *figure 8.2* montre de façon très visuelle le principe de fonctionnement de ce mode comparaison.

Un mode supplémentaire est disponible et s'appelle mode déclenchement spécial. Dans ce cas, une comparaison réussie provoque les événements suivants :

- le *timer 1* est remis à 0. Cela permet d'utiliser le registre CCPRx correspondant comme un registre de période programmable à 16 bits pour le *timer 1* ;
- dans certains circuits PIC, le *timer 1* est remis à 0 et une conversion analogique/digitale est déclenchée, si bien sûr la fonction correspondante est autorisée au niveau du convertisseur. Cela permet de réaliser très facilement des conversions à intervalles réguliers sans qu'il soit nécessaire de faire appel à l'unité centrale.

Notez que ce mode de déclenchement spécial ne positionne pas le bit d'indication d'interruption TMR1F dans le registre PIRx correspondant et ne génère donc pas d'interruption.

© Dunod. La photocopie non autorisée est un délit.

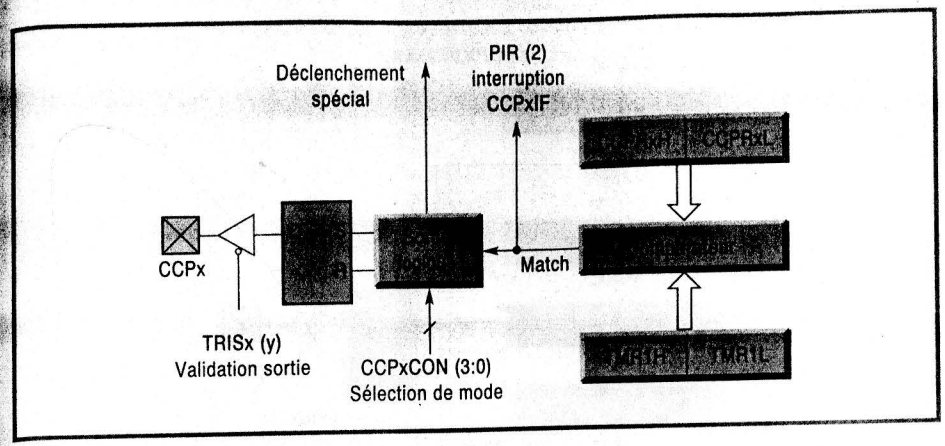


Figure 8.2 – Schéma équivalent du module de comparaison.

8.3 Le mode modulation de largeur d'impulsions ou mode PWM

Dans ce mode, la patte CCPx correspondant au module CCP choisi permet de disposer d'une sortie de signal modulé en largeur d'impulsions avec une résolution pouvant aller jusqu'à 10 bits. La patte correspondante du port parallèle avec laquelle elle est commune doit évidemment être configurée en sortie au moyen du bit adéquat du registre TRISx.

Comme le montre la *figure 8.3*, le principe de ce mode de fonctionnement est le suivant : l'utilisateur écrit dans le registre CCPxL le rapport cyclique désiré, donc codé sur 8 bits ; le registre CCPxH est utilisé comme esclave de CCPxL et ne peut donc être écrit par vos soins.

La période du signal généré quant à elle est déterminée par le contenu du registre période du *timer 2* (PR2). Dans ces conditions, et compte tenu du principe de fonctionnement du *timer 2* vu au chapitre 7, on peut écrire :

$$\text{Période du signal} = (PR2 + 1) \times 4T \times \text{Prédiviseur de TMR2}$$

où T est la période du signal d'horloge du PIC.

$$\text{Rapport cyclique} = DC1 \times T \times \text{Prédiviseur de TMR2}$$

T est la période d'horloge du PIC et *DC1* est le mot de 10 bits formé des 8 bits contenus dans CCPxL et des bits 4 et 5 de CCPxCON utilisés comme bits de poids faibles.

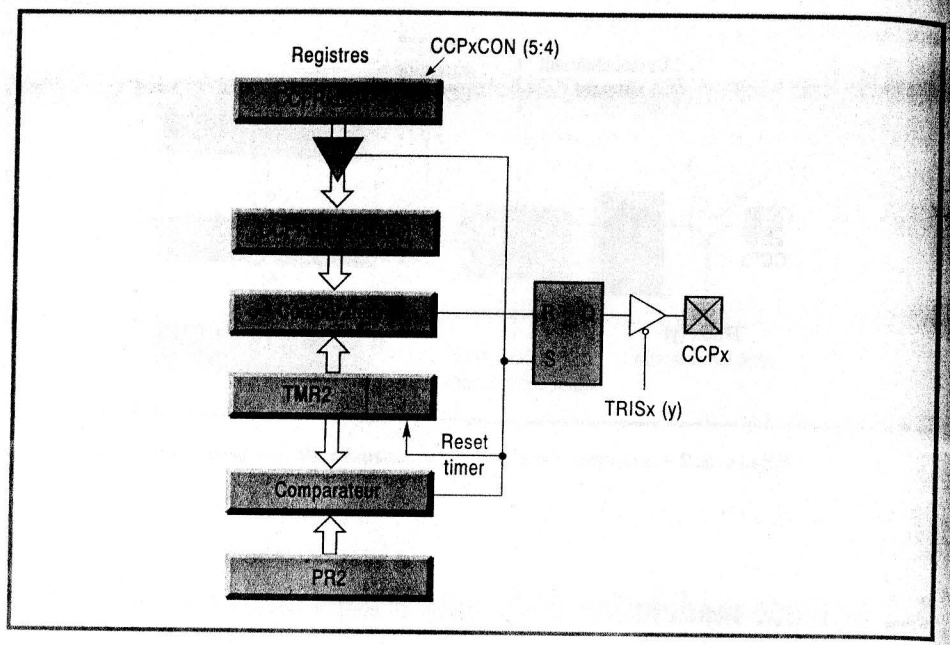


Figure 8.3 – Schéma équivalent du module PWM.

La résolution de ce mode PWM est donc bien programmable jusqu'à pouvoir atteindre 10 bits si ces bits 4 et 5 sont différents de 0. Dans le cas contraire, le mode PWM fonctionne en résolution sur 9 bits (un seul bit à zéro) ou 8 bits (les deux bits à zéro).

Compte tenu du principe de fonctionnement du mode PWM et des registres qui y sont utilisés, il est possible de dresser deux tableaux mettant en évidence diverses limites techniques. Le *tableau 8.4* présente tout d'abord la résolution minimale qu'il est possible d'obtenir en fonction de la programmation du prédiviseur du *timer 2*.

Le *tableau 8.5* quant à lui indique la résolution maximale qu'il est possible d'atteindre en fonction de la fréquence du signal PWM généré, son contenu ayant été calculé pour un PIC fonctionnant avec un quartz à 20 MHz (donc horloge système à 5 MHz).

Tableau 8.4 – Résolution minimum du mode PWM.

Prédiviseur	T2CKPS1	T2CKPS0	Résolution minimum
1	0	0	T_{OSC}
4	0	1	T_{CY}
16	1	X	$4T_{CY}$

Tableau 8.5 – Fréquence type et résolution du mode PWM pour une fréquence d'horloge de 20 MHz.

Fréquence (kHz)	1,22	4,88	19,53	78,12	153,6	208,3
Prédiviseur	16	4	1	1	1	1
Contenu de PR2	0xFF	0xFF	0xFF	0x3F	0x1F	0x17
Résolution maximale (nombre de bits)	10	10	10	8	7	5,5

8.4 Le registre de contrôle des modules CCP

Ces divers modes de fonctionnement étant vus, l'analyse du contenu du registre de contrôle de chaque module CCP va être facilitée, ce qui justifie le fait que nous ne vous la présentions que maintenant.

Chaque module CCP dispose de son propre registre de contrôle, appelé CCPxCON, dont chaque bit a la fonction que voici :

Accès E : Écriture L : Lecture	L	L	L/E	L/E	L/E	L/E	L/E	L/E
CCPxCON	—	—	DCxB1	DCxB0	CCPxM3	CCPxM2	CCPxM1	CCPxM0
État à la mise sous tension	0	0	0	0	0	0	0	0

- ***Bits 0 à 3 ou bits CCPxM0 à CCPxM3 pour CCPx Mode***

  Ces bits permettent de sélectionner le mode de fonctionnement en capture, comparaison ou PWM selon les indications du *tableau 8.6*, dans lequel on retrouve bien toutes les configurations que nous venons d'étudier.

- ***Bits 4 et 5 ou bits DCxB1 et DCxB0 pour Duty Cycle Bits***

  Ces deux bits ne sont utilisés qu'en mode PWM. Ils constituent les deux bits de poids faibles lorsque l'on veut travailler en mode résolution sur 10 bits. S'ils sont laissés à 0, le mode PWM fonctionne alors en résolution sur 8 bits ou sur 9 bits.

- ***Bits 6 et 7***

  Non utilisés et lus comme étant à 0.

Tableau 8.6 – Programmation des bits de mode de fonctionnement des modules CCP.

CCPxM3	CCPxM2	CCPxM1	CCPxM0	Mode
0	0	0	0	Arrêt tous modes
0	1	0	0	Capture tous les fronts descendants
0	1	0	1	Capture tous les fronts montants
0	1	1	0	Capture tous les 4 fronts montants
0	1	1	1	Capture tous les 16 fronts montants
1	0	0	0	Sortie à 1 si comparaison réussie
1	0	0	1	Sortie à 0 si comparaison réussie
1	0	1	0	Interruption si comparaison réussie
1	0	1	1	Mode déclenchement spécial
1	1	X	X	Mode PWM

© Dunod. La photocopie non autorisée est un délit.

CHAPITRE 9

LE PORT SÉRIE SYNCHRONE OU SSP

La généralisation des circuits à interface série synchrone, qu'elle soit de type général, Microwire, SPI, I2C ou toute autre appellation, a conduit très rapidement les fabricants de microcontrôleurs à pourvoir leurs circuits d'une telle ressource. Les microcontrôleurs PIC ne font pas exception à cette règle, puisque de nombreux boîtiers proposent une telle interface, appelée SSP pour *synchronous serial port*. Et comme les différentes interfaces séries de ce type, dont nous venons de citer les noms brièvement, reposent toutes sur le même principe et ne diffèrent que par des détails de protocole ou de procédure, le SSP des PIC est capable de s'adapter à n'importe laquelle d'entre-elles, vu sa souplesse de programmation. Pour cela, ce port dispose de deux modes de fonctionnement principaux.

Le premier mode se retrouve sur d'autres microcontrôleurs (avec la même appellation d'ailleurs) tels que, par exemple, les 68HC05 et 68HC11 de Motorola. Il est également compatible avec quasiment tous les circuits périphériques, mémoires et afficheurs LCD à accès série synchrone.

Le second mode permet l'interfaçage avec le bus I2C, créé à l'origine par Philips, mais adopté depuis par de très nombreux constructeurs. C'est donc une ressource très intéressante que nous offrent là les PIC qui en sont munis.

9.1 Le SSP en mode SPI

Lorsqu'il fonctionne en mode série synchrone de type SPI, le SSP permet la connexion, sous forme série maître-esclave, de plusieurs microcontrôleurs de la même famille ou de familles disposant d'interfaces compatibles et, bien entendu, la connexion de tous les périphériques ou mémoires munis d'une interface de ce type, appelée parfois interface série 2 ou 3 fils.

La *figure 9.1* présente le principe général d'interconnexion retenu. Ce schéma ne doit cependant pas être considéré comme unique. On peut en effet relier le SPI à

de très nombreux autres circuits : registres à décalage, afficheurs disposant de leur propre logique, etc. Il suffit de s'assurer qu'ils sont à même de dialoguer avec l'interface, ce qui est une contrainte relativement simple à satisfaire vu la souplesse de programmation permise.

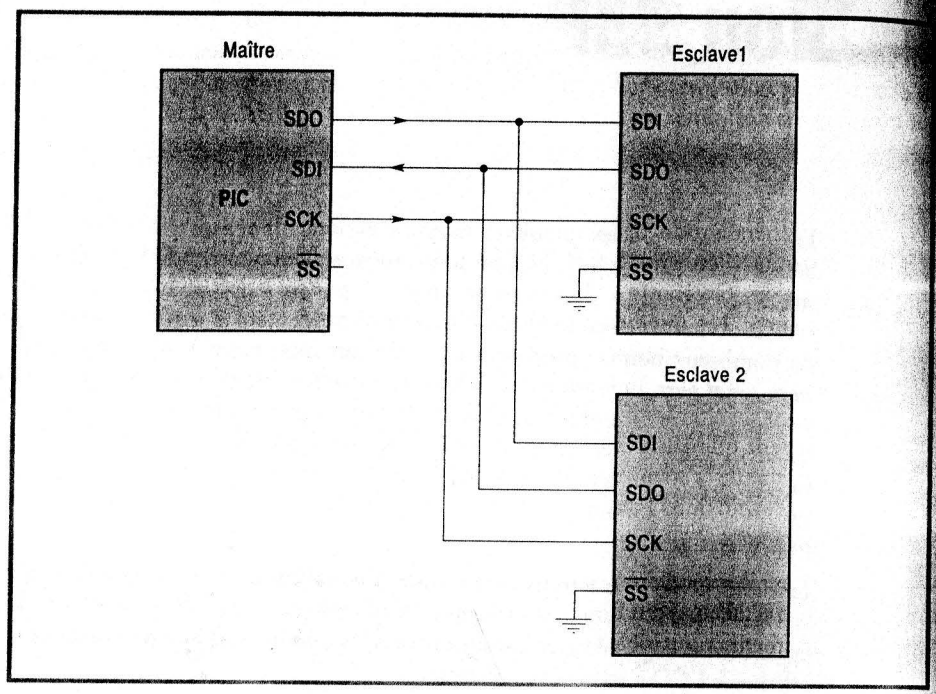


Figure 9.1 – Principe d'interconnexion de circuits dotés d'un port SPI.

Si l'on reprend la *figure 9.1* comme exemple, on y voit trois lignes « utiles » que sont SDO, SDI et SCK et une ligne de sélection du mode de fonctionnement du circuit concerné appelée $\overline{SS}$:

- La patte SDO pour *serial data out* est la sortie de données séries.
- La ligne SDI pour *serial data input* est, comme son nom le laisse supposer, l'entrée de données séries.
- La ligne SCK pour *serial clock* est utilisée pour synchroniser les échanges de données, puisque nous sommes ici en mode série synchrone ne l'oublions pas. Cette ligne est nécessairement une sortie sur le maître et une entrée sur les esclaves. La *figure 9.2* montre les relations de phase qui existent entre les données et cette horloge, étant entendu que celle-ci peut être programmée grâce au bit CKP que nous verrons dans un instant.

- La ligne $\overline{SS}$ est la sélection du mode maître ou esclave. Elle est au niveau haut sur un maître et au niveau bas sur un esclave. Pour éviter toute erreur de transmission, cette ligne doit être au niveau bas sur le ou les esclaves avant toute transmission, et y rester pendant toute la durée de celle-ci. En outre, il est évident qu'à un instant donné, n'importe quel circuit peut être maître et n'importe quel autre esclave, mais qu'il ne doit y avoir qu'un maître par transaction.

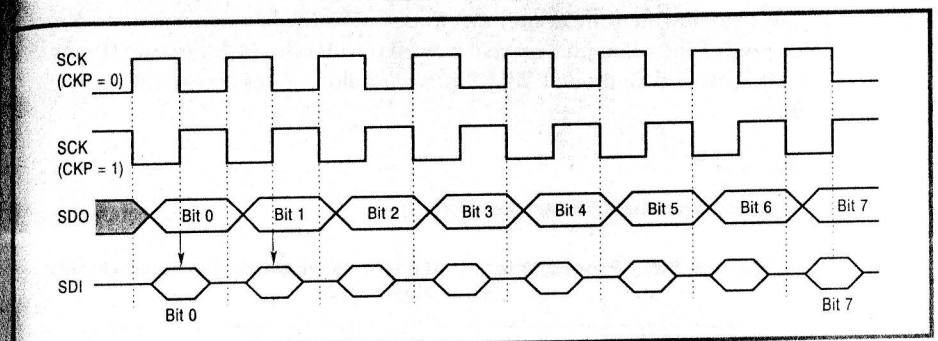


Figure 9.2 – Relation de phase entre données et horloge en mode SPI.

La *figure 9.3* présente le synoptique interne détaillé de l'interface série synchrone en mode SPI. Le registre principal est le registre SSPSR qui est en fait le registre à décalage d'émission et de réception.

Comme il est de coutume sur ce type d'interface, ce registre n'est pas directement accessible, mais seulement au travers du registre SSPBUF dans lequel sont écrites les données à émettre et où sont lues les données reçues. Cette façon de procéder permet de lire une donnée alors que la suivante est en cours de réception. Par contre, il est interdit d'écrire dans ce registre alors qu'une émission ou une réception est en cours. Si tel est le cas, le bit d'indication de collision WCOL est positionné dans le registre de contrôle présenté ci-après pour vous en informer.

Pour connaître l'état du registre SSPBUF, le bit BF du registre d'état du SSP est à votre disposition selon une procédure très classique. Ce bit peut être testé par logiciel, mais vous pouvez aussi travailler sous interruption, sous réserve d'avoir autorisée celle-ci par validation du bit adéquat dans le registre PIEx correspondant, comme nous l'avons vu au chapitre 3.

En résumé, pour utiliser cette interface en mode SPI, vous devez programmer un certain nombre de bits du registre de contrôle la concernant afin de :

- définir le mode maître ou le mode esclave ;
- définir la polarité de l'horloge, c'est-à-dire en fait définir si les données seront considérées comme valides sur un front montant ou descendant du signal d'horloge SCK (voir *figure 9.2*) ;

- définir la fréquence d'horloge (mode maître seulement) ;
- définir le mode de sélection de l'esclave (mode esclave seulement).

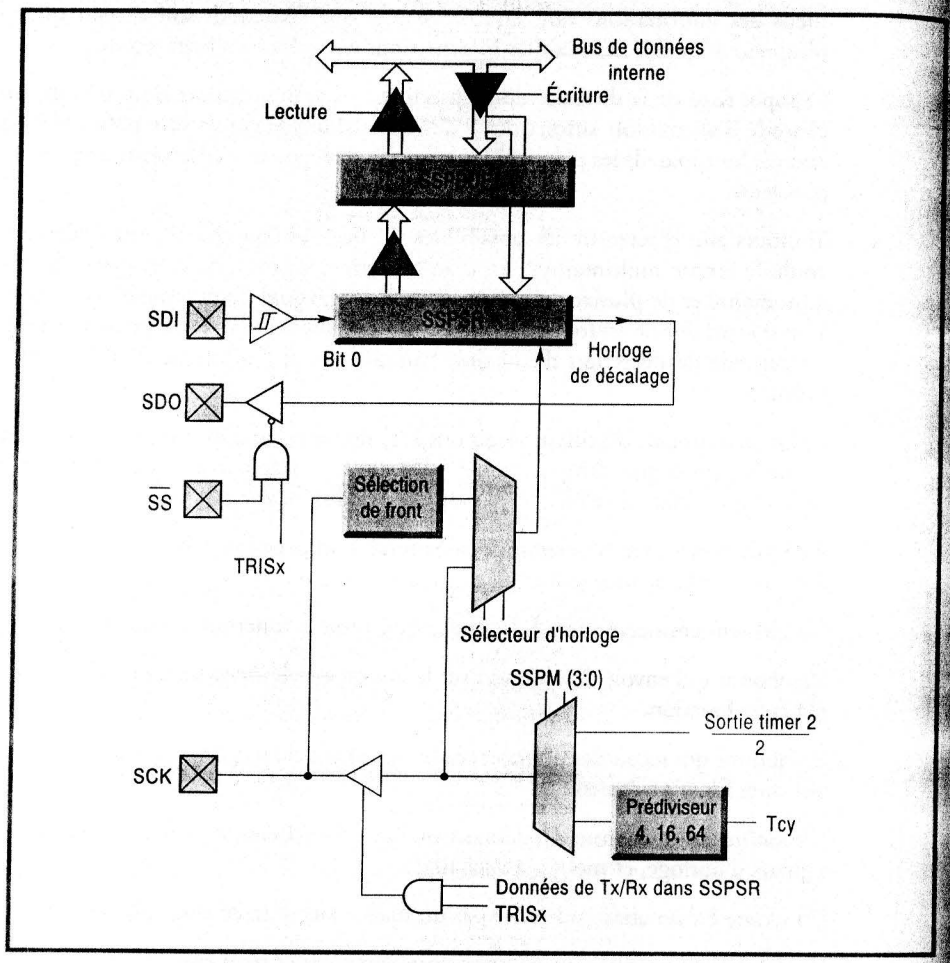


Figure 9.3 – Schéma équivalent du SSP en mode SPI.

Compte tenu du partage des lignes de cette interface avec celles d'un port parallèle, il faut aussi programmer correctement le registre TRISx correspondant, afin que les lignes partagées avec le SSP soient dans les sens de fonctionnement appropriés, à savoir que :

- SDI soit placée en entrée ;
- SDO soit placée en sortie ;

- SCK soit placée en sortie en mode maître et en entrée en mode esclave ;
- $\overline{SS}$ soit placée en entrée.

Si, compte tenu du mode de fonctionnement utilisé par cette interface, vous n'avez pas besoin de certaines lignes, comme par exemple dans le cas où un PIC en mode maître pilote un afficheur qui n'envoie jamais de données en retour, vous pouvez librement utiliser les lignes du port parallèle ainsi libérées.

Les chronogrammes des signaux rencontrés sur une liaison de type SPI sont visibles *figure 9.4* et *9.5* en mode maître ou esclave sans utilisation de $\overline{SS}$ et en mode esclave avec utilisation de $\overline{SS}$.

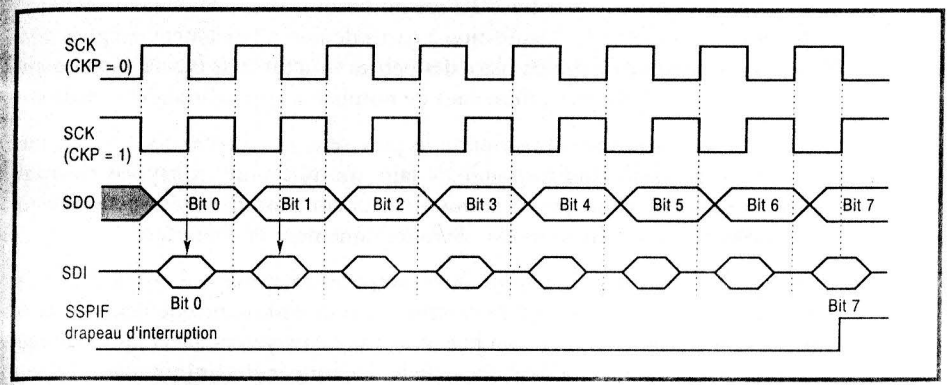


Figure 9.4 – Chronogrammes de transmission en mode maître SPI.

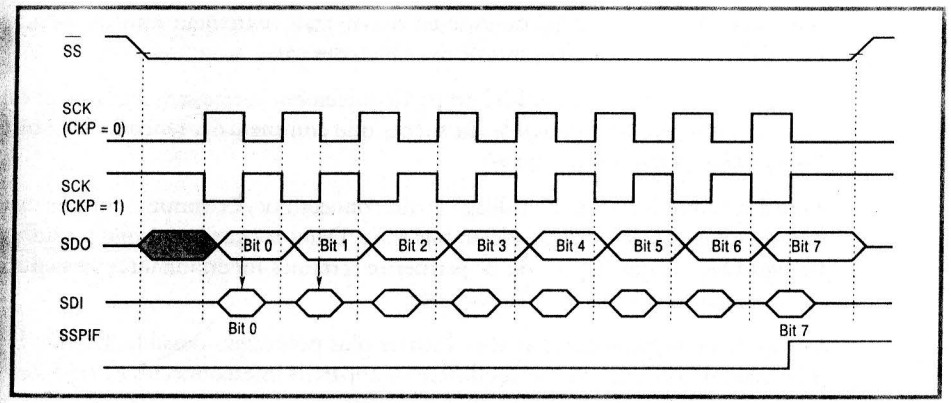


Figure 9.5 – Chronogrammes de transmission en mode esclave SPI.

La fréquence de l'horloge de transmission générée par le microcontrôleur est définie par des bits appropriés du registre de contrôle du port, présenté ci-après, et peut aller jusqu'à 5 MHz avec un boîtier exploitant la fréquence d'horloge maximum permise de 20 MHz.

9.2 Le SSP en mode I2C

Capable de relier jusqu'à 128 appareils divers au moyen de deux paires de fils torsadés très ordinaires, le bus I2C est un produit particulièrement intéressant, que ce soit pour faire de l'acquisition à partir de sites relativement éloignés les uns des autres ou pour mettre en place des petites structures de contrôle-commande. Ce bus est également très utilisé dans de nombreux appareils audio et vidéo.

Avant de voir comment fonctionne le port série synchrone des PIC en mode I2C, il nous semble indispensable de faire un minimum de rappels théoriques quant aux grands principes de ce bus, car leur connaissance est indispensable à la compréhension des divers modes de fonctionnement de l'interface.

Le bus I2C, dont le sigle signifie *inter integrated circuit* ce qui donne IIC et, par contraction, I2C, est un bus de communication n'utilisant que deux fils et une masse commune. C'est donc un bus série qui, au moyen de deux lignes de signal permet à un certain nombre d'appareils d'échanger des informations avec un débit pouvant atteindre 100 Kbits/s pour ce qui est de sa première mouture, et jusqu'à 400 Kbits/s pour les versions les plus récentes.

Les premières applications du bus I2C ont été des applications audio ou vidéo pour lesquelles la simplicité de mise en œuvre était nettement plus importante que le débit élevé, tout à fait inutile dans bien des cas.

Le cahier des charges du bus I2C est particulièrement intéressant à examiner car c'est lui le principal responsable du succès que commence à rencontrer ce bus. Voyons rapidement ce qu'il en est.

Afin de réduire les coûts de câblage et de connectique, et comme nous l'avons déjà dit ci-dessus, le bus I2C est un bus série. Deux fils auraient donc pu suffire (le signal et la masse) mais, afin de permettre certaines fonctionnalités particulières, une solution à trois fils a été adoptée.

Le bus doit être bidirectionnel afin d'être le plus polyvalent possible. En effet les informations à échanger entre les différents appareils interconnectés ne transitent pas nécessairement toujours dans le même sens.

Afin de pouvoir connecter de nombreux appareils sur ce bus et compte tenu de l'utilisation de deux fils « actifs » seulement, une notion d'adresse doit être prévue

afin que, lorsqu'un élément envoie une information, elle ne soit prise en compte que par le ou les appareils auxquels elle est réellement destinée.

Pour assurer un maximum de sécurité de transmission, un procédé d'acquittement des informations doit être prévu afin que l'émetteur soit certain que le récepteur auquel il destinait les informations envoyées les a bien reçues.

Compte tenu de la diversité des appareils pouvant être équipés d'un tel bus, un procédé d'adaptation automatique de vitesse doit également être prévu afin de ralentir les appareils les plus rapides lorsqu'ils dialoguent avec les équipements les plus lents.

Toujours afin d'accroître les possibilités du bus, les concepteurs ont également voulu le rendre multimaîtres. En d'autres termes, le bus I2C n'est pas composé d'un maître et de plusieurs esclaves, car n'importe quel élément peut être maître à un instant donné en fonction des besoins de l'application. Pour permettre cela, un procédé de détection d'émissions simultanées et d'arbitrage doit donc être prévu.

Enfin, le protocole d'utilisation du bus I2C doit être suffisamment bien défini et normalisé pour que n'importe quel fabricant de composants puisse réaliser des circuits d'interface à ce bus, ce qui est une condition *sine qua non* de succès.

Avant de poursuivre cet exposé, livrons-nous à un petit cours de vocabulaire afin que nous parlions tous le même langage.

Un élément connecté sur le bus, quelle qu'en soit la fonction, est un *abonné*.

Un abonné qui envoie des données sur le bus est appelé *émetteur*, tant qu'il accomplit cette fonction.

Un abonné qui reçoit des données sur le bus est appelé *récepteur*, tant qu'il accomplit cette fonction également.

Un *maître* est un abonné qui démarre ou initie un échange de données, génère les signaux d'horloge, et met fin à l'échange.

Un *esclave* est un abonné adressé par un maître lors d'un échange de données.

Un *échange* est un dialogue complet entre émetteur et récepteur.

Un *arbitrage* est une procédure permettant de résoudre un problème d'accès au bus lorsque deux maîtres veulent démarrer un transfert au même moment.

Ces notions doivent bien être perçues et surtout ne pas être confondues. De trop nombreuses personnes mélangent en effet la notion de maître et d'émetteur ainsi que celle d'esclave et de récepteur. Cela n'a rien à voir : ainsi dans un échange, un maître peut être récepteur et un esclave émetteur sans que cela représente la moindre ambiguïté.

Cela étant vu, voici quels sont les points forts du bus I2C :

- c'est un bus série bifilaire utilisant une ligne de données appelée SDA (*serial data*) et une ligne d'horloge appelée SCL (*serial clock*) ;
- les données peuvent être échangées dans les deux sens sans restriction ;
- le bus est multimaîtres ;
- chaque abonné dispose d'une adresse codée sur 7 bits. On peut donc connecter simultanément 128 abonnés d'adresses différentes sur le même bus (sous réserve de ne pas le surcharger électriquement bien sûr) ;
- un acquittement est généré pour chaque octet de donnée transféré ;
- le bus peut travailler à une vitesse maximum de 100 Kbits/s (400 Kbits/s pour certains circuits récents), étant entendu qu'un procédé automatique permet de ralentir l'équipement le plus rapide pour s'adapter à la vitesse de l'élément le plus lent lors d'un transfert.

Tout cela étant précisé, voyons maintenant comment fonctionne réellement ce bus car, à la lecture de ce qui précède et si vous avez quelques notions de micro-informatique et de bus plus traditionnels, vous devez commencer à vous poser bien des questions.

La *figure 9.6* montre le principe adopté au niveau des étages d'entrée/sortie des circuits d'interface au bus I2C. Si la partie entrée n'appelle aucune remarque particulière, on constate que la partie sortie fait appel à une configuration à drain ouvert (l'équivalent en MOS du classique collecteur ouvert), ce qui permet de réaliser des ET câblés par simple connexion, sur la ligne SDA ou SCL, des sorties de tous les circuits. Aucune charge n'étant prévue dans ces derniers, une résistance de rappel à une tension positive doit être prévue au moins une fois sur le bus.

Le niveau électrique n'étant pas précisé pour l'instant car il dépend de cette tension, nous parlerons de niveaux logiques haut ou 1 ou bien encore de niveaux logique bas ou 0, étant entendu que l'on travaille en logique positive, c'est-à-dire qu'un niveau haut correspond à une tension plus élevée qu'un niveau bas.

Compte tenu de ce mode de connexion en ET câblé, lorsqu'aucun abonné n'émet sur le bus, les lignes SDA et SCL sont au niveau haut qui est leur état de repos.

Cela étant vu, la *figure 9.7* résume le principe fondamental d'un transfert, à savoir : une donnée n'est considérée comme valide sur le bus que lorsque le signal SCL est à l'état haut. L'émetteur doit donc positionner la donnée à émettre lorsque SCL est à l'état bas, et la maintenir tant que SCL reste à l'état haut.

Comme la transmission s'effectue sous forme série, une information de début et de fin doit être prévue. L'information de début s'appelle ici condition de départ et l'information de fin condition d'arrêt.

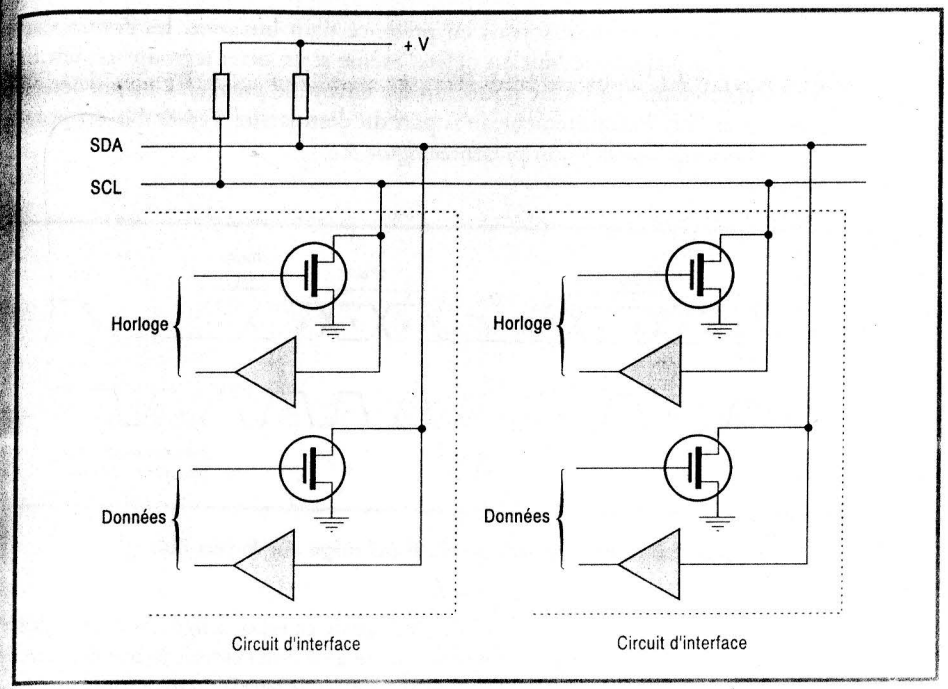


Figure 9.6 – Schéma équivalent des étages d'interface pour bus I2C.

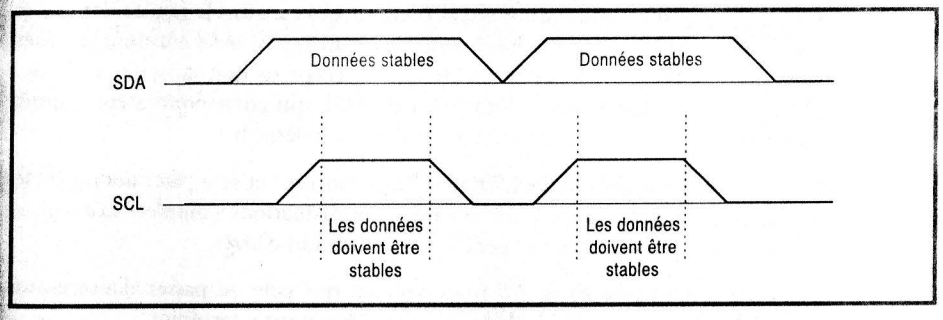


Figure 9.7 – Conditions de validité des données sur le bus I2C.

Une condition de départ est réalisée lorsque la ligne SDA passe du niveau haut au niveau bas alors que SCL est au niveau haut. Réciproquement, une condition d'arrêt est réalisée lorsque SDA passe du niveau bas au niveau haut alors que SCL est au niveau haut.

En outre, bien que nous soyons en présence d'un bus série, les données sont envoyées par paquets de huit ou octets, même si un octet regroupe en fait huit bits indépendants. Le bit de poids fort est envoyé le premier. Chaque octet est suivi par un bit d'acquittement de la part du destinataire, l'ensemble du processus fonctionnant de la façon présentée *figure 9.8.*

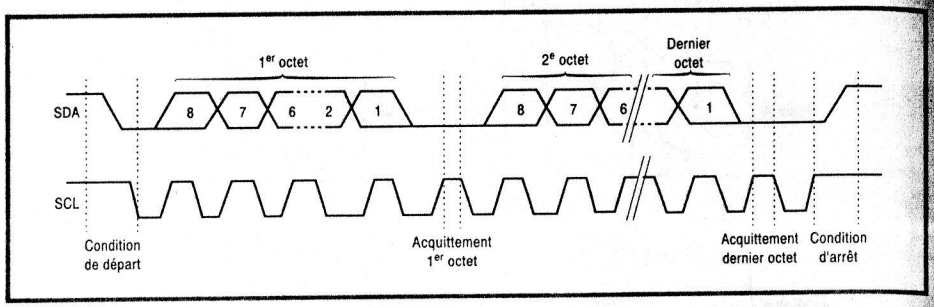


Figure 9.8 – Chronogramme d'un échange sur le bus I2C.

Précisons tout d'abord que, lors d'un échange de ce type, la ligne SCL est pilotée par l'initiateur de l'échange ou maître, quitte à ce que l'esclave agisse également dessus dans certains cas particuliers, comme nous le verrons dans un instant.

La *figure 9.8* nous montre tout d'abord une condition de départ, générée par le maître du bus à cet instant. Elle est suivie par le premier octet de données, poids forts en tête. Après le huitième bit, l'émetteur qui est aussi le maître dans ce cas, met sa ligne SDA au niveau haut c'est-à-dire au repos, mais continue à générer l'horloge sur SCL. Pour acquitter l'octet, le récepteur doit alors forcer la ligne SDA au niveau bas pendant l'état haut de SCL qui correspond à cet acquittement, prenant en quelque sorte la place d'un neuvième bit.

Le processus peut alors continuer avec l'octet suivant et se répéter autant de fois que nécessaire pour réaliser un échange d'informations complet. Lorsque cet échange est terminé, le maître génère une condition d'arrêt.

Utilisons toujours la *figure 9.8* pour voir ce qui peut se passer d'anormal au niveau du récepteur et ce que le bus I2C a prévu pour y remédier.

Supposons tout d'abord que, suite à la réception d'un octet, le récepteur ait besoin d'un certain temps pour exécuter l'ordre qu'on vient de lui transmettre et qu'il ne puisse donc accepter d'autres octets durant cette exécution. Il lui suffit, après avoir reçu cet octet, de maintenir sa ligne SCL au niveau bas pour que l'émetteur, détectant ce phénomène en raison de la structure en ET câblé présentée *figure 9.6*, suspende la transmission. Dès que la ligne SCL remonte, l'émetteur reprend le transfert.

Si, maintenant, le récepteur ne peut acquitter la donnée reçue, soit parce qu'il l'a mal reçue, soit encore parce qu'il est occupé à autre chose ; l'émetteur, ne voyant pas arriver cet acquittement, génère alors une condition d'arrêt de façon à interrompre cet échange raté.

Nous venons de voir là les deux seuls cas de problèmes de transmission pouvant se produire lors d'un échange. Voyons maintenant ceux qui peuvent avoir lieu avant un échange. Cela se produit en général lorsque deux abonnés veulent initialiser un échange au même moment, c'est-à-dire lorsqu'ils veulent tous deux devenir maîtres du bus.

Si cette situation se produit, deux conflits différents risquent d'avoir lieu : celui qui vient à l'esprit en premier concerne bien évidemment les données qui sont mutuellement perturbées mais, en fait, le conflit commence au niveau de la ligne d'horloge puisque nous avons vu que SCL était pilotée par l'initiateur de l'échange. Deux processus différents sont donc mis en œuvre.

Le premier est celui de la synchronisation des horloges sur la ligne SCL. En effet, comme le montre la *figure 9.9*, et du fait de la structure en ET câblé du bus, l'horloge adopte automatiquement la configuration suivante :

– état bas aussi long que le plus long des états bas en cours de génération ;

– état haut aussi court que le plus court des états hauts en cours de génération.

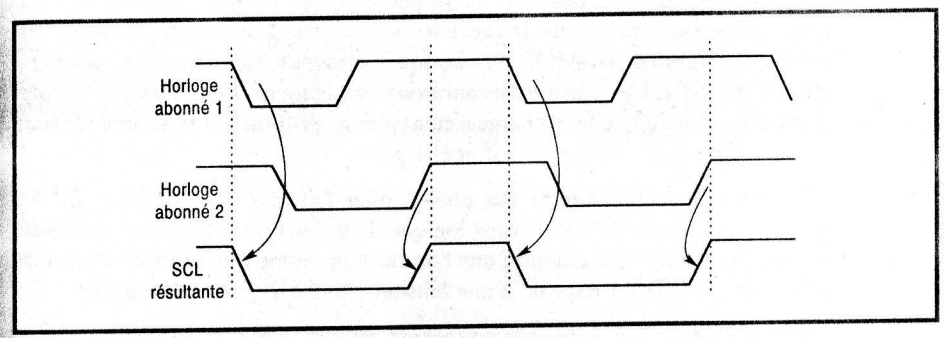


Figure 9.9 – Principe de synchronisation de l'horloge.

Comme chaque circuit relit en permanence ses propres informations de sortie, cette synchronisation des horloges réalise une synchronisation des maîtres potentiels du bus.

Intéressons nous maintenant aux données en examinant la *figure 9.10.* Toujours en raison de la relecture immédiate des données envoyées en sortie, dès que deux circuits tentent d'envoyer simultanément sur la ligne SDA des données différentes, celui qui a émis un niveau haut se rend immédiatement compte d'une anomalie

puisque, du fait de la structure en ET câblé, le niveau bas est prioritaire. Il considère alors qu'un autre abonné a pris la maîtrise du bus à sa place et il abandonne l'échange en libérant sa ligne SDA, c'est-à-dire en la plaçant au niveau haut.

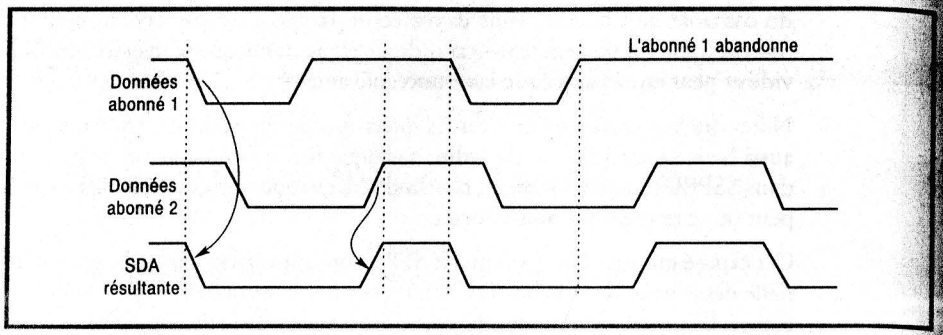


Figure 9.10 – Principe d'arbitrage en cas de conflit d'accès au bus.

Nous savons maintenant comment se déroulent les échanges et le règlement des différents conflits pouvant se produire sur le bus, il nous reste à examiner le format des données transmises afin de comprendre comment fonctionne l'adressage, mais aussi la définition du sens de transfert des données.

La *figure 9.11* montre le contenu du premier octet qui est toujours présent en début d'échange. Ses 7 bits de poids forts contiennent l'adresse du destinataire du message, ce qui autorise 128 combinaisons différentes. Le bit de poids faible quant à lui indique si le maître va réaliser une lecture ou une écriture. En d'autres termes, si ce bit est à 0, le maître va écrire dans l'esclave ou lui envoyer des données. Si ce bit est à 1, le maître va lire dans l'esclave, c'est-à-dire que le maître va recevoir des données de l'esclave.

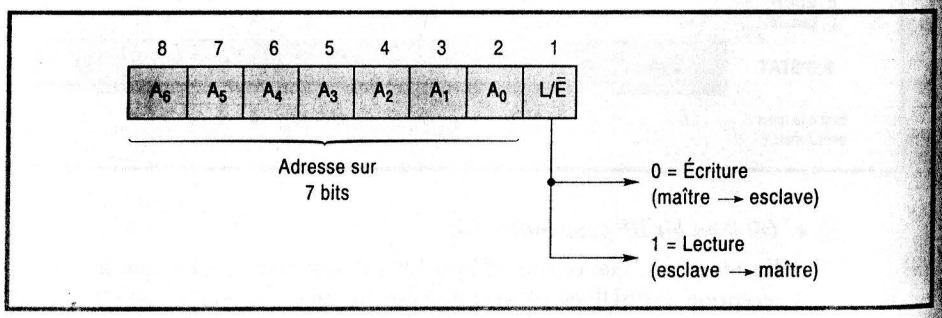


Figure 9.11 – Format du premier octet en mode adressage sur 7 bits.

C'est cette situation particulière qui nous a amenés à écrire, en début d'exposé, qu'il ne fallait pas confondre maître et émetteur ou bien encore esclave et récepteur. En effet, dans ce cas le maître est récepteur et l'esclave est émetteur, alors que c'est tout de même le maître qui a été l'initiateur de l'échange et qui en assure le contrôle par la ligne SCL et la vérification des acquittements.

Cet adressage sur 7 bits n'étant plus suffisant dans certaines situations, un adressage sur 10 bits a récemment été introduit et fait appel à deux octets pour coder l'adresse comme indiqué *figure 9.12*. Le poids fort de l'adresse ainsi qu'un code particulier et le bit de lecture écriture sont envoyés dans le premier octet, tandis que le second contient le poids faible de cette adresse. Cela ne change rien bien évidemment aux principes exposés ci-dessus.

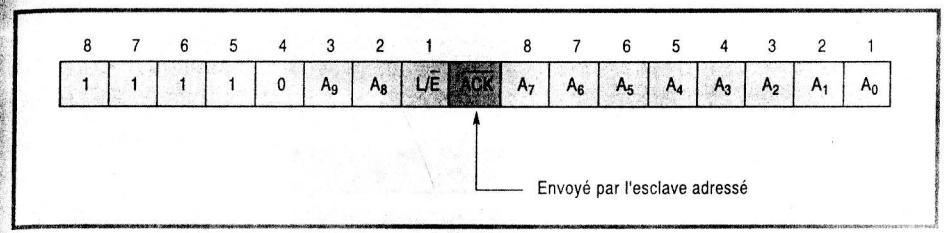


Figure 9.12 – Format des deux premiers octets en mode adressage sur 10 bits.

Pour fonctionner en mode I2C, le SSP adopte l'architecture interne visible *figure 9.13* et fait donc appel à cinq registres :

- le registre de contrôle SSPCON ;
- le registre d'état SSPSTAT ;
- le registre buffer d'émission/réception SSPBUF ;
- le registre à décalage d'émission/réception SSPSR qui n'est pas directement accessible comme dans le mode SPI ;
- le registre d'adresse SSPADD.

Une fois programmé en mode I2C grâce aux bits adéquats du registre de contrôle, le SSP est capable de réaliser automatiquement les fonctions d'un esclave du bus I2C en adressage 7 bits ou 10 bits. Il peut aussi fonctionner en mode maître, mais nécessite dans ce cas un plus gros travail logiciel.

Pour fonctionner en mode esclave, les deux lignes du port parallèle qui véhiculent SDA et SCL doivent être configurées en entrées grâce aux bits appropriés du registre TRISx correspondant. Le SSP se chargera ensuite de forcer en sortie telle ou telle de ces lignes lorsque ce sera rendu nécessaire par le protocole.

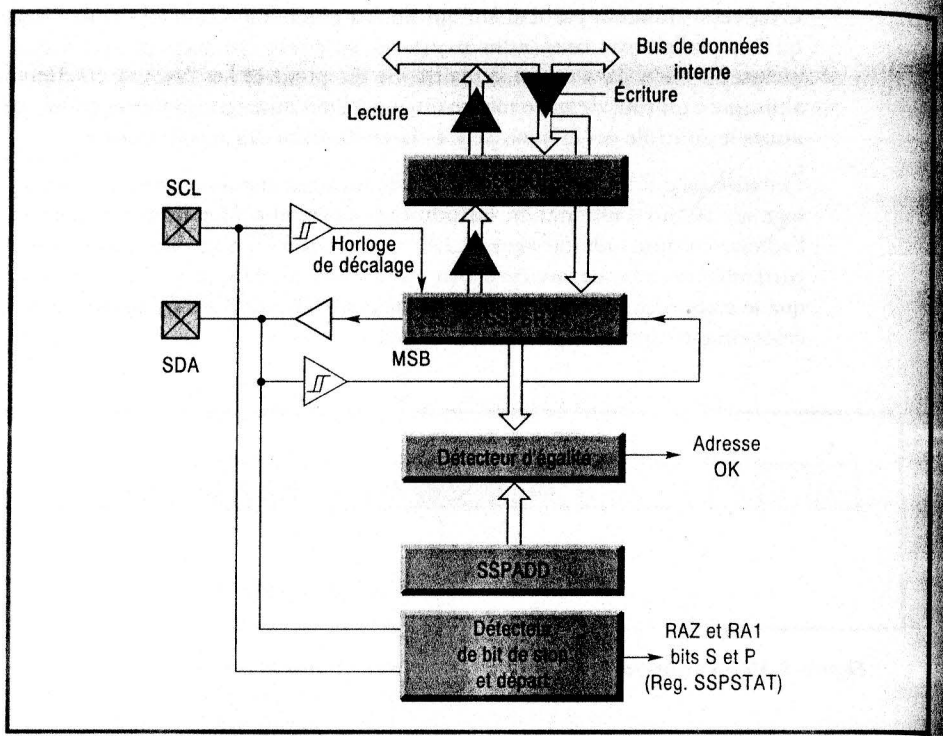


Figure 9.13 – Schéma équivalent du SSP en mode I2C.

Le principe de fonctionnement en réception est le suivant. Lors de la réception d'un premier octet valide, il est comparé au contenu du registre SSPADD (en mode adresse 7 bits). S'il y a égalité, le registre SSPBUF est chargé avec l'octet reçu, le bit d'indication de registre SSPBUF plein est positionné, une interruption est générée si elle y a été autorisée par le bit approprié du registre PIEx, et l'impulsion $\overline{\text{ACK}}$ propre à l'I2C est générée automatiquement par action sur SDA et SCL.

En mode adresse 10 bits, le même principe reste valable mais, du fait du découpage de l'adresse en deux octets, une intervention par logiciel est nécessaire. En effet, le processus ci-dessus se produit pour l'octet de poids fort de l'adresse. S'il y a égalité de cet octet avec celui contenu dans SSPADD, il faut alors charger l'octet de poids faible de cette adresse dans ce registre afin qu'il puisse être comparé au deuxième octet reçu sur le bus I2C.

La réception des données se poursuit alors normalement avec, à chaque octet valide, un positionnement du bit du registre d'état indiquant que le registre SSPBUF contient une donnée à lire. La génération de l'$\overline{\text{ACK}}$ sur le bus I2C est

automatique ainsi que sa non-génération dans le cas, par exemple, où vous n'avez pas lu assez rapidement le contenu de SSPBUF et où il risque donc d'y avoir écrasement des données.

Le fonctionnement en émission est aussi simple. Après la phase de réception et de contrôle de l'adresse, vous devez écrire la première donnée à émettre dans SSPBUF. Un bit du registre d'état indique ici aussi lorsque le registre SSPBUF est vide et peut être chargé avec la donnée suivante.

Notez que vous pouvez autoriser les interruptions émanant du SSP en mode I2C aussi bien en réception, c'est-à-dire à chaque fois qu'un octet valide est contenu dans SSPBUF, qu'en émission, c'est-à-dire à chaque fois que SSPBUF est vide et peut donc recevoir un nouvel octet.

Cet exposé montre donc bien que le SSP assure automatiquement la gestion matérielle des échanges sur le bus I2C sans nécessiter le recours de l'unité centrale. Par contre, la gestion de fonctions plus élaborées telles que la détection de collision ou le fonctionnement en mode maître demande bien évidemment l'écriture d'un programme approprié.

9.3 Les registres du SSP

Maintenant que nous connaissons les deux modes de fonctionnement de l'interface série synchrone ou SSP, nous pouvons examiner le rôle des divers bits d'état et de contrôle de ses registres internes. Voyons tout d'abord le registre d'état SSPSTAT dont chaque bit a la signification particulière que voici :

Accès E : Écriture L : Lecture	L/E	L/E	L	L	L	L	L	L
SSPSTAT	SMP	CKE	D/$\overline{\text{A}}$	P	S	R/$\overline{\text{W}}$	UA	BF
État à la mise sous tension	0	0	0	0	0	0	0	0

- ***Bit 0 ou bit BF pour buffer full***

  En réception, que ce soit SPI ou I2C, ce bit passe à 1 lorsque le registre de réception SSPBUF est plein et doit être lu. En émission, SPI ou I2C, ce bit est à 1 tant qu'une émission a lieu et qu'il ne faut donc pas écrire dans SSPBUF. Il passe à 0 en fin d'émission lorsque la donnée suivante peut être placée dans SSPBUF.

© Dunod. La photocopie non autorisée est un délit.

- ***Bit 1 ou bit UA pour update address***

  Ce bit ne sert qu'en mode I2C à adressage sur 10 bits. Il passe à 1 lorsqu'il faut charger le registre SSPADD avec l'octet de poids faible de l'adresse.

- ***Bit 2 ou bit R/$\overline{W}$ pour read/write***

  Ce bit n'a de signification qu'en mode I2C et indique, s'il est à 1, que l'on est en présence d'une lecture. Lorsqu'il est à 0, il signale une écriture. Ce bit n'a de signification que pendant le temps qui s'écoule entre la reconnaissance d'une adresse valide et la condition d'arrêt ou de départ suivante.

- ***Bit 3 ou bit S pour start bit***

  Ce bit n'a de signification qu'en mode I2C et passe à 1 lorsque la détection d'une condition de départ a eu lieu.

- ***Bit 4 ou bit P pour stop bit***

  Ce bit n'a de signification qu'en mode I2C et passe à 1 lorsque la détection d'une condition d'arrêt a eu lieu.

- ***Bit 5 ou bit D/$\overline{A}$ pour data/address***

  Ce bit n'a de signification qu'en mode I2C et passe à 1 lorsque le dernier octet émis ou reçu est une donnée. Il reste à 0 si c'est une adresse.

- ***Bit 6 ou bit CKE pour clock edge select***

  Ce bit n'a de signification qu'en mode SPI. Lorsqu'il est à 1, il indique qu'une donnée est transmise sur un front montant du signal SCK et lorsqu'il est à 0, qu'elle est transmise sur un front descendant. Cela étant valide pour CKP à 0 dans le registre SSCON présenté ci-dessous. Si CKP de ce même registre est à 1, les indications du bit CKE sont inversées (1 pour front descendant et 0 pour front montant).

- ***Bit 7 ou bit SMP pour sample phase***

  Lorsque le SSP fonctionne en mode SPI maître, ce bit indique, lorsqu'il est à 1, que les données sont échantillonnées à la fin de la période de données. S'il est à 0, les données sont échantillonnées au centre de la période de données. Lorsque le SSP fonctionne en mode SPI esclave, ce bit doit rester à 0.

À la lumière des explications que nous vous avons fournies sur les principes de fonctionnement en mode SPI ou I2C, on constate donc que le contenu de ce registre est fort logique. En mode SPI, seul le bit BF est utilisé, alors qu'en mode I2C l'examen des autres bits de ce registre permet de suivre avec une bonne précision ce qui se passe sur la liaison.

© Dunod. La photocopie non autorisée est un délit.

La configuration du SSP dans un mode ou l'autre a lieu grâce au registre de contrôle dont chaque bit a la signification particulière que voici :

Accès E : Écriture L : Lecture	L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E
SSPCON	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0
État à la mise sous tension	0	0	0	0	0	0	0	0

- ***Bits 0 à 3 ou bits SSPM0 à SSPM3 pour SSPMode 0 à 3***

  Ces bits permettent de sélectionner le mode de fonctionnement du SSP selon les indications fournies par le *tableau 9.1* ci-dessous.

- ***Bit 4 ou bit CKP pour clock polarity select***

  En mode SPI, les données sont valides sur un front descendant de l'horloge si ce bit est à 1 et sur un front montant s'il est à 0. En mode I2C la ligne SCL fonctionne normalement si ce bit est à 1, alors qu'elle est maintenue à 0 si ce bit est à 0.

- ***Bit 5 ou bit SSPEN pour SSP enable***

  En mode SPI, le SSP est validé lorsque ce bit est à 1 et les lignes SCK, SDO, SDI et SS barre sont donc actives. Le SSP est invalidé si ce bit est à 0 et les lignes correspondantes du port parallèle peuvent donc être utilisées pour un usage général. En mode I2C, le principe est le même, mais pour ce qui est des lignes du port parallèle supportant SDA et SCL.

- ***Bit 6 ou bit SSPOV pour SSP overflow***

  En mode SPI comme en mode I2C, ce bit passe à 1 lorsqu'une nouvelle donnée est reçue alors que le contenu précédent du registre SSPBUF n'a pas encore été lu. Le programme de gestion du SSP doit lire le registre SSPBUF, même s'il ne fait que de l'émission de données, afin d'éviter le positionnement à 1 de ce bit. Par ailleurs, ce bit doit être remis à 0 par logiciel.

- ***Bit 7 ou bit WCOL pour write collision detect***

  Ce bit est mis à 1 lors d'une tentative d'écriture dans le registre SSPBUF alors qu'il est en train de transmettre l'octet précédent. Ce bit doit être remis à 0 par logiciel.

Le *tableau 9.1* présente les divers modes de fonctionnement du SSP tels qu'ils sont codés au moyen des bits SSPM du registre de contrôle.

Tableau 9.1 – Programmation des bits SSPM0 à SSPM3 de sélection de mode de fonctionnement du SSP.

$SSPM_3$	$SSPM_2$	$SSPM_1$	$SSPM_0$	Mode SSP
0	0	0	0	SPI - Maître - Horloge = $F_{osc}/4$
0	0	0	1	SPI - Maître - Horloge = $F_{osc}/16$
0	0	1	0	SPI - Maître - Horloge = $F_{osc}/64$
0	0	1	1	SPI - Maître - Horloge = Sortie TMR_2
0	1	0	0	SPI - Esclave - $\overline{SS}$ valide
0	1	0	1	SPI - Esclave - $\overline{SS}$ invalide
0	1	1	0	I^2C - Esclave - Adresse 7 bits
0	1	1	1	I^2C - Esclave - Adresse 10 bits
1	0	1	1	I^2C - Maître
1	1	1	0	I^2C - Maître/Esclave - Adresse 7 bits
1	1	1	1	I^2C - Maître/Esclave - Adresse 10 bits

CHAPITRE 10

L'INTERFACE DE COMMUNICATION SÉRIE OU SCI

L'interface de communication série ou SCI pour *serial communication interface* dont sont équipés un certain nombre de PIC est en fait ce que l'on appelle plus communément un USART, c'est-à-dire un émetteur/récepteur universel de données synchrones et asynchrones. De fait, la SCI peut fonctionner comme une interface série asynchrone classique en *full duplex* mais aussi comme une interface série synchrone, restreinte dans ce cas au seul mode *half duplex*.

Son mode d'utilisation le plus courant reste bien évidemment le mode série asynchrone permettant ainsi la connexion de l'application dans laquelle elle est utilisée à la majorité des équipements informatiques classiques.

La vitesse de transmission de la SCI est totalement programmable grâce à un registre interne, mais est identique en émission et réception. La parité n'est pas supportée directement par la SCI en ce sens qu'elle ne la calcule ni ne la vérifie. Par contre, elle offre la possibilité de transmettre et de recevoir 9 bits de données, ce qui permet de gérer cette parité par logiciel et d'utiliser ce neuvième bit pour l'émettre et la recevoir.

Hormis cette programmation sur 8 bits ou 9 bits, le format de transmission est fixe mais classique avec un bit de *start*, les bits de données et un bit de stop.

10.1 La SCI en mode asynchrone *full duplex*

Dans ce mode de fonctionnement, la partie émission de la SCI adopte le synoptique visible *figure 10.1*. Il reste très classique puisque le cœur de cet ensemble est le registre à décalage d'émission qui n'est pas directement accessible en écriture. On doit en effet écrire la donnée à transmettre dans le registre TXREG lorsqu'un bit du registre d'état associé nous indique que ce registre est vide. Cette situation peut être testée par lecture de ce bit par logiciel ou générer une interruption si la SCI y a été autorisée grâce au bit approprié du registre PIEx (voir chapitre 3 si nécessaire).

© Dunod. La photocopie non autorisée est un délit.

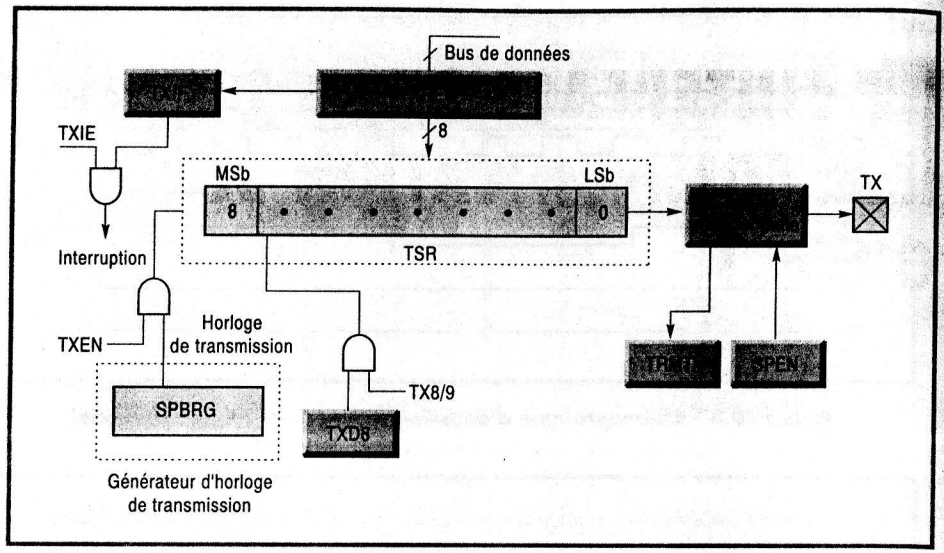


Figure 10.1 – Schéma équivalent de la SCI en mode émission asynchrone.

Si la transmission doit être faite sur 9 bits (utilisation de la parité par exemple avec une transmission de données sur 8 bits), ce neuvième bit provient du registre TXSTA que nous verrons dans un instant. Il doit être positionné correctement avant d'écrire dans le registre TXREG. En effet, si TXREG est vide, le fait d'y écrire un octet valide immédiatement la transmission ; le neuvième bit doit donc impérativement avoir été positionné au préalable.

Le principe d'utilisation de la SCI en émission est donc le suivant :

- écriture de la vitesse de transmission désirée dans le registre SPBRG (présenté ci-après) ;
- validation du port SCI et configuration correcte du sens de travail des pattes du port parallèle avec lequel la SCI partage ses lignes d'émission de données TX/CK et de réception de données RX/DT par positionnement des bits du registre TRIS correspondant ;
- validation éventuelle des interruptions en émission par positionnement du bit TXIE du registre PIEx concerné ;
- test du bit TRMT du registre TXSTA (présenté ci-après) pour voir si le registre d'émission est vide, sauf en cas de fonctionnement sous interruption où le déclenchement de l'interruption correspondante signale cette situation ;
- en cas d'émission de 9 bits, positionnement correct du neuvième bit dans le registre TXSTA ;

© Dunod. La photocopie non autorisée est un délit.

- écriture de l'octet à transmettre dans le registre TXREG.

Le processus se répète ensuite au niveau du test de l'état du registre TXREG pour la transmission des données suivantes.

En réception, le principe de fonctionnement est encore très classique, la SCI adoptant alors la structure visible *figure 10.2.*

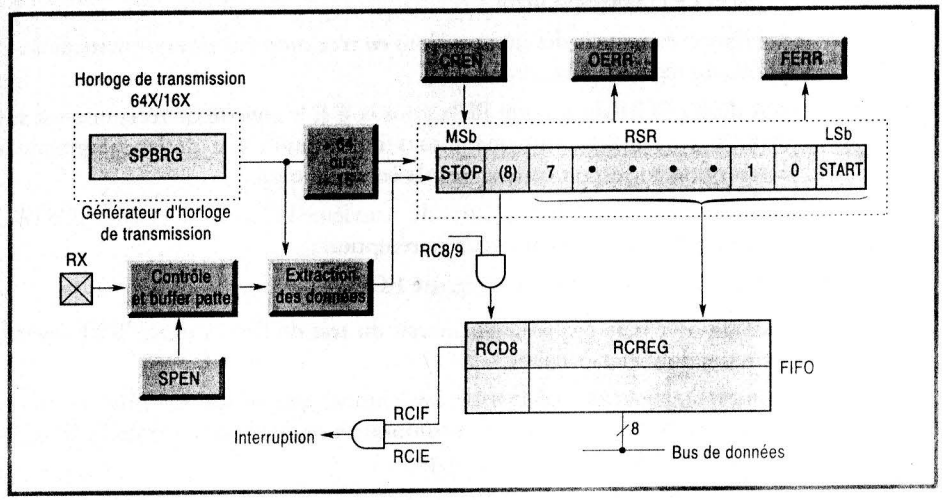


Figure 10.2 – Schéma équivalent de la SCI en mode réception asynchrone.

Comme il se doit, le registre à décalage de réception n'est pas accessible directement et ne peut être lu que par l'intermédiaire du registre RCREG qui dispose d'un double buffer ou pile FIFO (*first in, first out* pour premier entré, premier sorti). Il est donc possible de recevoir un octet alors que le précédent n'a pas encore été lu.

Un bit du registre d'état indique bien évidemment quant un octet est disponible dans RCREG et peut provoquer une interruption sous réserve, d'y avoir été autorisé par positionnement du bit adéquat dans le registre PIEx.

Deux types de contrôles sont en outre disponibles dans le registre d'état. Un bit de débordement qui signale l'écrasement des données contenues dans RCREG par les données suivantes si elles n'ont pas été lues assez vite, et un bit d'erreur de format. Aucun bit d'erreur de parité n'est prévu, puisque cette dernière n'est pas directement supportée.

Le neuvième bit, s'il est utilisé, n'est pas disponible dans RCREG qui est un registre 8 bits classique, mais dans le registre d'état de réception comme nous le verrons lors de sa description.

Le principe d'utilisation de la SCI en réception est donc le suivant :

- écriture de la vitesse de transmission désirée dans le registre SPBRG (présenté ci-après) ;
- validation du port SCI et configuration correcte du sens de travail des pattes du port parallèle avec lequel la SCI partage ses lignes d'émission de données TX/CK et de réception de données RX/DT par positionnement des bits du registre TRIS correspondant ;
- validation éventuelle des interruptions en réception par positionnement du bit RCIE du registre PIEx concerné ;
- test du bit RCIF du registre PIRx pour voir si le registre de réception est vide sauf en cas de fonctionnement sous interruption où le déclenchement de l'interruption correspondante signale cette situation ;
- en cas de réception 9 bits, lecture du neuvième bit dans le registre RCSTA et vérification de l'absence d'erreur de réception ;
- lecture de l'octet reçu dans le registre RCREG.

Le processus se répète ensuite au niveau du test de l'état du bit RCIF pour la réception des données suivantes.

Ce fonctionnement en mode série asynchrone, que ce soit en émission ou en réception, est donc très classique et conforme à ce que l'on rencontre habituellement sur toutes les interfaces de ce type.

10.2 La SCI en mode synchrone *half duplex*

D'utilisation plus rare, le mode synchrone est aussi proposé par la SCI des PIC, mais il ne peut alors fonctionner qu'en *half duplex*. Son principe global reste identique à celui vu ci-dessus pour le mode asynchrone avec toutefois deux différences majeures :

- s'agissant d'un mode *half duplex*, l'émission et la réception simultanées ne sont pas possibles. Le fait d'utiliser une de ces fonctions désactive donc automatiquement l'autre ;
- le fait d'être maintenant en mode synchrone signifie que l'horloge de transmission est envoyée conjointement aux données. Lorsque la SCI est en mode maître, c'est donc elle qui fournit cette horloge sur sa patte CK, à une fréquence fixée comme dans le cas précédent par le contenu du registre SPBRG. Lorsque la SCI est en mode esclave, elle reçoit l'horloge de transmission sur cette même patte CK.

La *figure 10.3* précise les chronogrammes d'une émission en mode maître, tandis que la *figure 10.4* indique ceux d'une réception dans ce même mode.

© Dunod. La photocopie non autorisée est un délit.

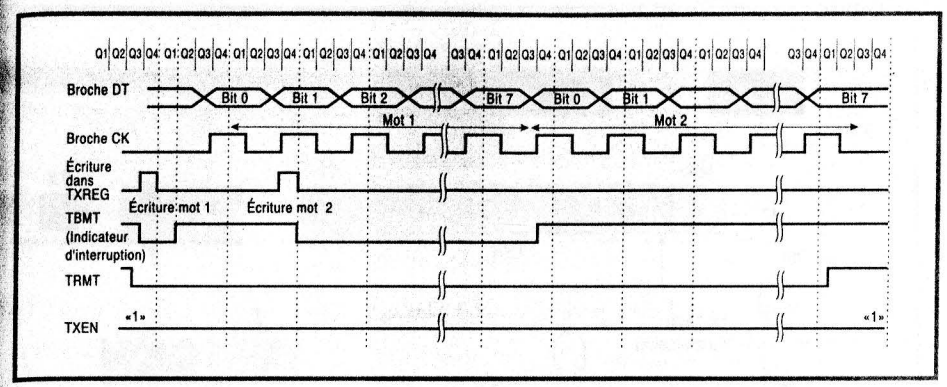


Figure 10.3 – Chronogramme d'émission de la SCI en mode synchrone.

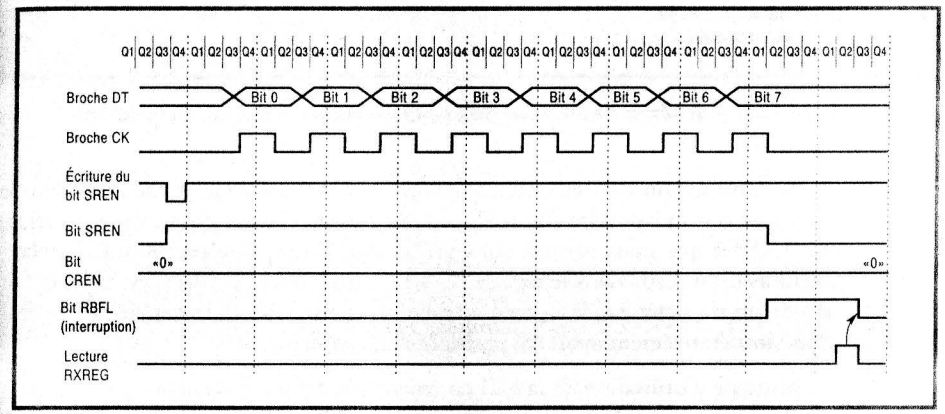


Figure 10.4 – Chronogramme de réception de la SCI en mode synchrone.

Les principes d'émission et réception vus ci-dessus pour le mode asynchrone restent valables. Il suffit juste de positionner correctement les bits de sélection de ce mode dans les registres TXSTA et RCSTA.

10.3 Les registres de la SCI

Maintenant que nous connaissons les deux modes de fonctionnement de l'interface de communication série, nous pouvons examiner le rôle des divers bits d'état et de contrôle de ses registres internes. Voyons tout d'abord le registre d'état TXSTA dont chaque bit a la signification particulière que voici :

Accès E : Écriture L : Lecture	L/E	L/E	L/E	L/E	L	L/E	L	L/E
TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D
État à la mise sous tension	0	0	0	0	0	0	1	0

- ***Bit 0 ou bit TX9D pour Transmit Data 9***

  Ce bit n'est autre que le neuvième bit de donnée dans le cas d'une émission sur 9 bits.

- ***Bit 1 ou bit TRMT pour TRansMit shift registrer empTy***

  Lorsque ce bit est à 1, le registre d'émission TXREG est vide et peut donc recevoir une nouvelle donnée.

- ***Bit 2 ou bit BRGH pour baud rate generator high***

  Ce bit sélectionne un des deux groupes de vitesses de transmissions permises par le registre SPBRG (voir ci-après). Il n'est utilisé qu'en mode asynchrone.

- ***Bit 3***

  Non utilisé et lu comme étant à zéro.

- ***Bit 4 ou bit SYNC pour synchronous mode***

  Ce bit doit être mis à 1 en mode synchrone et laissé à 0 dans le cas contraire.

- ***Bit 5 ou bit TXEN pour transmit enable***

  Ce bit doit être mis à 1 pour autoriser les émissions de données. Les bits SREN et CREN du registre RCSTA sont prioritaires sur ce bit en mode synchrone.

- ***Bit 6 ou bit TX9 pour transmit 9***

  Ce bit sélectionne une transmission sur 9 bits s'il est mis à 1 et sur 8 bits dans le cas contraire.

- ***Bit 7 ou bit CSRC pour clock source select***

  Ce bit est inutilisé en mode asynchrone. En mode synchrone il impose l'utilisation de l'horloge interne générée par SPBRG (mode maître) lorsqu'il est à 1, ou une horloge externe appliquée sur l'entrée CK (mode esclave) lorsqu'il est à 0.

Le deuxième registre principal contrôlant la SCI est le registre RCSTA dont chaque bit a la signification que voici :

© Dunod. La photocopie non autorisée est un délit.

Accès E : Écriture L : Lecture	L/E	L/E	L/E	L/E	L	L	L	L
RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D
État à la mise sous tension	0	0	0	0	0	0	0	0

- ***Bit 0 ou bit RXD9 pour receive data 9***

  Ce bit n'est autre que le neuvième bit de donnée en mode réception sur 9 bits.

- ***Bit 1 ou bit OERR pour overrun error***

  Ce bit passe à 1 en cas d'erreur de débordement, c'est-à-dire de lecture pas assez rapide du registre de réception RCREG. Il est remis à 0 par mise à 0 du bit CREN.

- ***Bit 2 ou bit FERR pour format error***

  Ce bit passe à 1 en cas d'erreur de format. Il est remis à 0 par une lecture du registre RCREG et la réception de la donnée suivante si elle est valide.

- ***Bit 3***

  Non utilisé et lu comme étant à zéro.

- ***Bit 4 ou bit CREN pour continuous receive enable***

  En mode asynchrone, ce bit doit être mis à 1 pour autoriser la réception. En mode synchrone il autorise la réception continue s'il est mis à 1. Ce bit est prioritaire sur le bit SREN.

- ***Bit 5 ou bit SREN pour single receive enable***

  En mode synchrone, ce bit doit être mis à 1 pour autoriser la réception, sauf en mode esclave où il est ignoré. En mode asynchrone, ce bit n'est pas utilisé.

- ***Bit 6 ou bit RX9 pour receive 9 bits***

  S'il est mis à 1, ce bit sélectionne une réception sur 9 bits, et sur 8 bits dans le cas contraire.

- ***Bit 7 ou bit SPEN pour serial port enable***

  Ce bit doit être mis à 1 pour valider les lignes RX/DT et TX/CK comme lignes du port série SCI et pour valider ce port.

Le dernier registre de contrôle de la SCI est le registre SPBRG, qui permet de définir la vitesse de transmission. Le contenu de ce registre permet en fait de calculer la vitesse de transmission qui sera utilisée selon les indications du *tableau 10.1*.

Tableau 10.1 – Mode de calcul de la vitesse de la SCI.

Bit SYNC	BRGH = 0	BRGH = 1
0	Vitesse = $F_{OSC}/(64(X+1))$	Vitesse = $F_{OSC}/(16(X+1))$
1	Vitesse = $F_{OSC}/(4(X+1))$	Non autorisé

En mode asynchrone, défini par la mise à 0 du bit SYNC du registre TXSTA, deux formules de calcul sont utilisables selon l'état du bit BRGH de ce même registre.

En mode synchrone, défini par la mise à 1 de ce même bit SYNC, une seule formule est utilisable, le bit BRGH n'étant pas exploité dans ce mode.

Dans les relations indiquées *tableau 10.1*, le nombre figuré par un X est le contenu du registre SPBRG et peut donc varier de 00 à FF, soit de 0 à 255. Le *tableau 10.2* donne quelques valeurs types dans différentes configurations et pour diverses fréquences d'horloge, ainsi que les pourcentages d'erreur obtenus par ce procédé par rapport aux vitesses théoriques.

Notez aussi que les circuits PIC suivants : 16C65, 65A, 73, 73A, 74 et 74A présentent un défaut de conception qui leur interdit d'être utilisés avec BRGH du registre TXSTA au niveau 1. La fréquence d'horloge générée alors n'est pas conforme à la valeur calculée et interdit toute utilisation correcte de la SCI. Il importe donc de n'utiliser ces circuits qu'avec BRGH au niveau 0.

© Dunod. La photocopie non autorisée est un délit.

Tableau 10.2 – Quelques vitesses de transmission de la SCI pour diverses fréquences d'horloges et différents contenus de SPBRG.

Bits/s (K)	f_{osc} = 20 MHz KBaud	% erreur	Valeur SPBRG (décimale)	f_{osc} = 16 MHz KBaud	% erreur	Valeur SPBRG (décimale)	f_{osc} = 10 MHz KBaud	% erreur	Valeur SPBRG (décimale)	f_{osc} = 7,15909 MHz KBaud	% erreur	Valeur SPBRG (décimale)
0,3	NA	-	-	NA	-	-	NA	-	-	NA	-	-
1,2	1,221	+1,73	255	1,202	+0,16	207	1,202	+0,16	129	1,203	+0,16	92
2,4	2,404	+0,16	129	2,404	+0,16	103	2,404	+0,16	64	2,380	+0,16	46
9,6	9,469	- 1,36	32	9,615	+0,16	25	9,766	+1,73	15	9,322	+0,16	11
19,2	19,53	+1,73	15	19,23	+0,16	12	19,23	+1,73	7	18,64	+0,16	5
76,8	78,13	+1,73	3	83,33	+8,51	2	68,13	+1,73	1	NA	+8,51	-
96	104,2	+8,51	2	NA	-	-	NA	-	-	NA	-	-
300	312,5	+4,17	0	NA	-	-	NA	-	-	NA	-	-
500	NA	-	-	NA	-	-	NA	-	-	NA	-	-
HIGH	312,5	-	0	250	-	0	156,3	-	0	111,9	-	0
LOW	1,221	-	255	0,977	-	255	0,6104	-	255	0,437	-	255

Bits/s (K)	f_{osc} = 5,0688 MHz KBaud	% erreur	Valeur SPBRG (décimale)	f_{osc} = 3,579545 MHz KBaud	% erreur	Valeur SPBRG (décimale)	f_{osc} = 1 MHz KBaud	% erreur	Valeur SPBRG (décimale)	f_{osc} = 32,768 kHz KBaud	% erreur	Valeur SPBRG (décimale)
0,3	0,31	+3,13	255	0,301+0,23+0,16		185	0,300	+0,16	51	0,256	- 14,67	1
1,2	1,2	0	65	1,190	- 0,83	46	1,202	+0,16	12	NA	-	-
2,4	2,4	0	32	2,432	+1,32	22	2,232	- 6,99	6	NA	-	-
9,6	9,9	+3,13	7	9,322	- 2,90	5	NA	-	-	NA	-	-
19,2	19,8	+3,13	3	18,64	- 2,90	2	NA	-	-	NA	-	-
76,8	79,2	+3,13	0	NA	-	-	NA	-	-	NA	-	-
96	NA	-	-	NA	-	-	NA	-	-	NA	-	-
300	NA	-	-	NA	-	-	NA	-	-	NA	-	-
500	NA	-	-	NA	-	-	NA	-	-	NA	-	-
HIGH	79,2	-	0	55,93	-	0	15,63	-	0	0,512	-	0
LOW	0,3094	-	255	0,2185	-	255	0,0610	-	255	0,0020	-	255

BRGH = 0

Bits/s (K)	f_{osc} = 20 MHz KBaud	% erreur	Valeur SPBRG (décimale)	f_{osc} = 16 MHz KBaud	% erreur	Valeur SPBRG (décimale)	f_{osc} = 10 MHz KBaud	% erreur	Valeur SPBRG (décimale)	f_{osc} = 7,16 MHz KBaud	% erreur	Valeur SPBRG (décimale)
9600	9615,34	+0,16	129	9615,34	+0,16	103	9615,39	+0,16	64	9520,07	- 0,83	46
19200	19230,77	+0,16	64	19230,77	+0,16	51	18939,39	-1,36	32	19454,05	+1,32	22
38400	37878,79	- 1,36	32	38461,54	+0,16	25	39062,50	+1,7	15	37286,93	- 2,90	11
57600	56818,18	- 1,36	21	58823,53	+2,12	16	56818,18	-1,36	10	55930,39	- 2,90	7
115200	113636,4	- 1,36	10	111111,1	- 3,55	8	125000	+8,51	4	111860,8	- 2,90	3
250000	250000	0	4	250000	0	3	NA	-	-	NA	-	-
625000	625000	0	1	NA	-	-	625000	0	0	NA	-	-
1250000	1250000	0	0	NA	-	-	NA	-	-	NA	-	-

	f_{osc} = 5,068 MHz KBaud	% erreur	Valeur SPBRG (décimale)	f_{osc} = 3,579 MHz KBaud	% erreur	Valeur SPBRG (décimale)	f_{osc} = 1 MHz KBaud	% erreur	Valeur SPBRG (décimale)	f_{osc} = 32,768 kHz KBaud	% erreur	Valeur SPBRG (décimale)
9600	9600	0	32	9727,02	+1,32	22	8928,57	- 6,99	6	NA	-	-
19200	18645,29	- 2,94	16	18643,46	- 2,90	11	20833,33	+8,51	2	NA	-	-
38400	39600	+3,12	7	37286,93	- 2,90	5	31250	- 18,61	1	NA	-	-
57600	52800	- 8,33	5	55930,39	- 2,90	3	62500	+8,51	0	NA	-	-
115200	105600	- 8,33	2	111860,8	- 2,90	1	NA	-	-	NA	-	-
250000	NA	-	-	223721,6	- 10,51	0	NA	-	-	NA	-	-
625000	NA	-	-	NA	-	-	NA	-	-	NA	-	-
1250000	NA	-	-	NA	-	-	NA	-	-	NA	-	-

BRGH = 1

CHAPITRE 11

LA SOURCE DE TENSION DE RÉFÉRENCE ET LES COMPARATEURS

Ces deux fonctions, assez peu fréquentes sur un microcontrôleur, sont indépendantes l'une de l'autre. Nous avons néanmoins décidé de les traiter dans un même chapitre car on les associe très souvent. En effet, comme son nom l'indique, la source de tension de référence est en fait une source de tension programmable au moyen d'un registre interne. Les comparateurs quant à eux sont en fait un double comparateur analogique, disposant de nombreux modes de configuration. Ils peuvent bien sûr comparer les niveaux de plusieurs tensions externes distinctes, mais peuvent aussi utiliser comme élément de comparaison la tension provenant de la source de tension de référence. Comme nous allons le voir sans plus tarder, et après les ressources très « riches » qu'étaient les SSP et SCI, la source de tension de référence et les comparateurs sont d'utilisation très simple.

11.1 La source de tension de référence

La source de tension de référence est constituée par un réseau de seize résistances internes alimenté par une fraction plus ou moins grande de la tension d'alimentation. Un multiplexeur analogique à seize entrées permet de prélever une tension en l'un quelconque des seize points de connexion des résistances afin de la rendre disponible pour le ou les comparateurs et pour une patte de sortie. La *figure 11.1* concrétise cela au moyen d'un schéma synoptique.

Comme on peut le constater sur cette figure, le réseau de résistances peut être alimenté sous la quasi totalité de la tension d'alimentation du PIC, lorsque le transistor commandé par le bit VRR du registre VRCON que nous verrons dans un instant est conducteur. Dans le cas contraire, la résistance 8R est mise en circuit et réduit la plage de tension d'alimentation qui alimente le réseau de résistances. On dispose donc de deux plages de tensions programmables qui vont de :

- 0 à 0,75 V_{DD} avec un pas égal à $V_{DD}/24$;
- 0,25 V_{DD} à 0,75 V_{DD} avec un pas égal à $V_{DD}/32$.

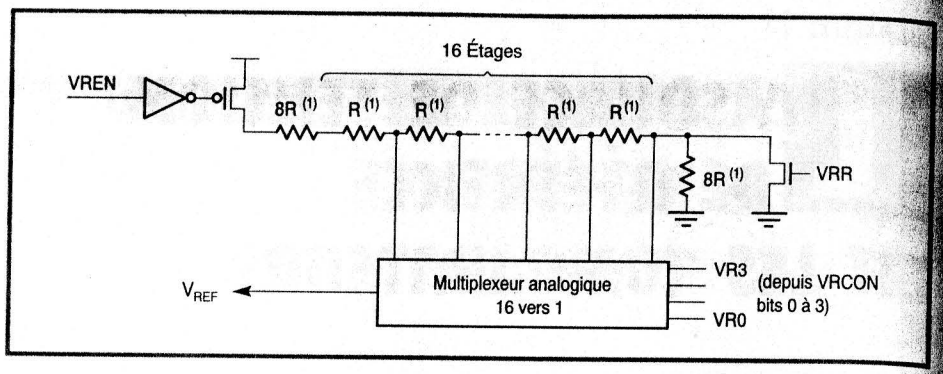


Figure 11.1 – Schéma équivalent de la source de tension de référence.

Compte tenu du mode de fonctionnement de cette source de tension, il est évident que sa précision ne saurait être supérieure à celle de la tension d'alimentation V_{DD} du PIC puisque c'est une fraction de cette dernière qui alimente le réseau de résistances. Il convient d'en tenir compte dans les applications où une grande précision est nécessaire.

Cette ressource est entièrement contrôlée par un seul registre appelé VRCON dont voici le contenu détaillé :

Accès E : Écriture L : Lecture	L/E	L/E	L/E	L	L/E	L/E	L/E	L/E
VRCON	VREN	VROE	VRR	—	VR3	VR2	VR1	VR0
État à la mise sous tension	0	0	0	0	0	0	0	0

- ***Bits 0 à 3 ou bits VR0 à VR3 pour voltage reference***

  Ces bits commandent le multiplexeur analogique et sélectionnent donc la valeur de la tension de référence, conformément aux indications du *tableau 11.1.*

- ***Bit 4***

  Non utilisé et lu comme étant à 0.

- ***Bit 5 ou bit VRR pour VR_{EF} range selection***

  Lorsqu'il est à 1, ce bit sélectionne la plage de fonctionnement de 0 à $0{,}75\,V_{DD}$. Il sélectionne la plage $0{,}25\,V_{DD}$ à $0{,}75\,V_{DD}$ dans le cas contraire.

- ***Bit 6 ou bit VROE pour VR_{EF} output enable***

  Lorsque ce bit est à 1, la sortie de la source de tension de référence est reliée à l'entrée du module comparateur. Elle est également disponible sur la patte V_{REF} du boîtier. Lorsque ce bit est à 0, la sortie de la source de tension de référence est déconnectée du module comparateur et de la patte V_{REF}.

- ***Bit 7 ou bit VREN pour VR_{EF} enable***

  Le générateur de tension de référence n'est alimenté que lorsque ce bit est mis à 1. Il est arrêté et ne consomme aucun courant dans le cas contraire.

La tension de référence est donc fonction des bits VR0 à VR3, qui sélectionnent le point de prélèvement de celle-ci sur le réseau de résistances, et du bit VRR qui permet de choisir sa plage de tension d'alimentation, comme nous l'avons vu sur le schéma synoptique. Le *tableau 11.1* résume toutes les combinaisons disponibles pour une tension d'alimentation de 5 V.

Pour une tension V_{DD} différente de 5 V, les relations à utiliser pour déterminer la tension de référence sont les suivantes :

– si le bit VRR est à 1 : $V_{REF} = (VR3{:}VR0/24)\,V_{DD}$, où VR3:VR0 est la valeur des bits VR3 à VR0 exprimée en décimal ;

– si le bit VRR est à 0 : $V_{REF} = V_{DD}/4 + (VR3{:}VR0/32)\,V_{DD}$, où VR3:VR0 est la valeur des bits VR3 à VR0 exprimée en décimal.

Ces relations peuvent évidemment être écrites autrement afin de déterminer le contenu des bits VR3 à VR0. On a alors :

– si le bit VRR est à 1 : $VR3{:}VR0 = 24\,V_{REF}/V_{DD}$;

– si le bit VRR est à 1 : $VR3{:}VR0 = 32(V_{REF} - V_{DD}/4)/V_{DD}$.

Notez que le contenu du registre VRCON n'est pas affecté par le passage en mode SLEEP ou par la sortie de ce mode suite à un réveil au moyen du *timer* chien de garde. Si vous souhaitez économiser l'énergie en mode SLEEP, il est donc nécessaire de désactiver par programme la tension de référence en mettant à 0 le bit 7 de VRCON.

11.2 Les comparateurs de tension

Les comparateurs de tension sont en fait constitués de deux comparateurs indépendants qui partagent leurs pattes d'entrées avec celles de ports parallèles, et qui peuvent utiliser ou non la source de tension de référence interne que nous venons de décrire. En pratique, ces comparateurs peuvent recevoir un total de huit configurations différentes présentées *figure 11.2.*

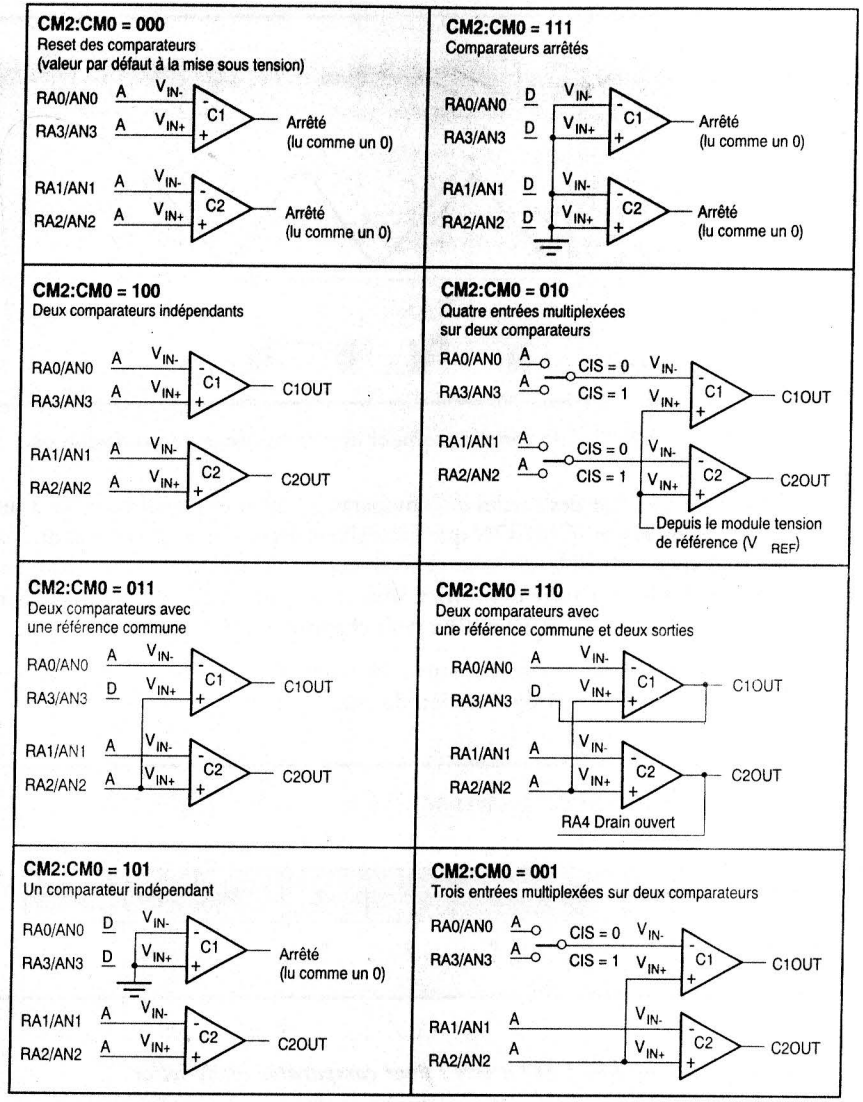
Tableau 11.1 – Les différentes tensions disponibles pour V_{DD} = 5 V.

				V_{REF}	
VR3	VR2	VR1	VR0	VRR = 1	VRR = 0
0	0	0	0	0,00	1,25
0	0	0	1	0,21	1,41
0	0	1	0	0,42	1,56
0	0	1	1	0,63	1,72
0	1	0	0	0,83	1,88
0	1	0	1	1,04	2,03
0	1	1	0	1,25	2,19
0	1	1	1	1,46	2,34
1	0	0	0	1,67	2,50
1	0	0	1	1,88	2,66
1	0	1	0	2,08	2,81
1	0	1	1	2,29	2,97
1	1	0	0	2,50	3,13
1	1	0	1	2,71	3,28
1	1	1	0	2,92	3,44
1	1	1	1	3,13	3,59

Comme vous pouvez le constater à l'examen de cette figure, on dispose de tous les modes souhaitables permettant de réaliser la comparaison de deux à quatre tensions externes, ou bien encore la comparaison de une ou plusieurs tensions externes avec la tension de référence interne.

Ces comparateurs fonctionnent comme de véritables comparateurs analogiques puisque, comme le montre la *figure 11.3*, leurs sorties passent à l'état haut lorsque la tension sur V_{IN+} est supérieure à celle sur V_{IN-} et vice versa. Comme dans un comparateur analogique « classique », une zone d'incertitude existe, repérée par les bandes noires larges sur la figure, en raison de la tension de décalage (*offset*) des comparateurs et de leur vitesse de réaction.

Dunod - La photocopie non autorisée est un délit



A = entrée analogique
D = entrée numérique
CIS = CMCON bit 3 (commutateur d'entrées)

Figure 11.2 – Schémas équivalents et modes de configuration des comparateurs analogiques.

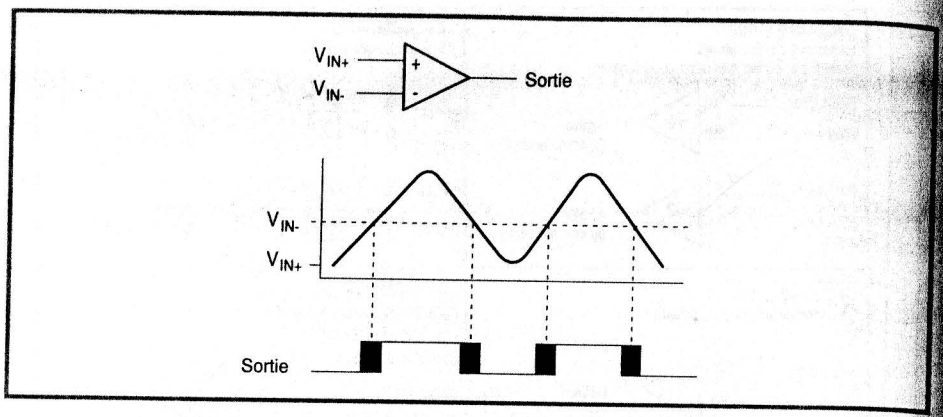


Figure 11.3 – Principe de fonctionnement des comparateurs analogiques.

La lecture de l'état des sorties des comparateurs peut être réalisée grâce à un bit d'un registre appelé CMCON que nous allons découvrir dans un instant, mais il est également possible de faire générer une interruption lors du changement d'état de l'une ou l'autre des sorties, sous réserve de positionner correctement le bit correspondant du registre PIEx (voir chapitre 3 si nécessaire).

Le fonctionnement du comparateur est contrôlé par un seul registre appelé CMCON dont voici la signification du contenu :

Accès E : Écriture L : Lecture	L	L	L	L	L/E	L/E	L/E	L/E
CMCON	C2OUT	C1OUT	—	—	CIS	CM2	CM1	CM0
État à la mise sous tension	0	0	0	0	0	0	0	0

- ***Bits 0 à 2 ou bits CM0 à CM2 pour comparator mode select***

  Ces bits permettent de sélectionner le mode de connexion des comparateurs, conformément aux schémas que vous avez pu découvrir en *figure 11.2*, sur laquelle était également indiqué le contenu des bits CM0 à CM2.

- ***Bit 3 ou bit CIS pour comparator input switch***

  Ce bit commande le commutateur utilisé dans les divers modes à entrées multiplexées. Il agit sur la position du ou des commutateurs d'entrées comme indiqué en *figure 11.2*, sur laquelle était également indiquée la valeur de ce bit.

- ***Bits 4 et 5***

  Non utilisés et lus comme étant à 0.

- ***Bit 6 ou bit C1OUT pour comparator 1 output***

  Ce bit est le reflet de l'état de la sortie du comparateur 1. Il est donc à 1 lorsque V_{IN1+} est supérieure à V_{IN1-} et à 0 dans le cas contraire.

- ***Bit 7 ou bit C2OUT pour comparator 2 output***

  Ce bit est le reflet de l'état de la sortie du comparateur 2. Il est donc à 1 lorsque V_{IN2+} est supérieure à V_{IN2-} et à 0 dans le cas contraire.

Compte tenu du partage des entrées/sorties des comparateurs avec des lignes de ports parallèles (RA0 à RA3 sur la majorité des circuits), le registre TRIS correspondant doit avoir ses bits correctement positionnés en fonction du mode choisi pour les comparateurs afin de permettre leur fonctionnement normal.

Notez à ce propos que la lecture d'un port parallèle dont les entrées sont utilisées comme entrées analogiques des comparateurs donne un résultat nul, alors que l'application d'une tension analogique à un port configuré comme entrée numérique peut conduire l'étage d'entrée de ce port à consommer un courant anormalement important.

Lorsque les interruptions issues du ou des comparateurs sont autorisées, le programme de traitement de ces interruptions doit exécuter la séquence suivante afin de remettre à zéro les interruptions en provenance des comparateurs :

– lecture ou écriture du registre CMCON, ce qui a pour effet de positionner les bits CM1OUT et CM2OUT à leurs nouvelles valeurs ;

– mise à zéro du drapeau d'interruption (bit CMIF) dans le registre PIRx correspondant.

Lorsqu'un comparateur est validé alors que le circuit est mis en mode SLEEP, le comparateur reste actif et peut générer une interruption s'il y a été autorisé. Si cette interruption se produit, elle fait alors sortir le circuit du mode SLEEP.

Si les comparateurs sont inutiles en mode SLEEP, il est conseillé de les arrêter en positionnant à 111 les bits CM0 à CM2 afin de minimiser la consommation du PIC. Notez à ce propos que le fait d'entrer ou de sortir du mode SLEEP ne modifie pas le contenu du registre CMCON.

Terminons cette présentation des comparateurs avec la *figure 11.4*, qui montre le schéma équivalent de chacune de leurs entrées. Compte tenu de la présence de diodes de protection reliées à V_{SS} et à V_{DD}, comme sur toutes les lignes d'entrées/sorties numériques des PIC, les tensions externes appliquées aux entrées des comparateurs doivent toujours être comprises entre V_{SS} et V_{DD}, et un système de limitation de courant doit être prévu si elles sont susceptibles de sortir

de cette plage. Par ailleurs, la mise en conduction d'une des diodes de protection peut provoquer un phénomène de *latch-up* ou verrouillage de l'entrée correspondante. Cette situation doit donc être évitée autant que faire se peut.

Compte tenu des caractéristiques de ces entrées analogiques, l'impédance de la source à laquelle elles sont reliées doit être inférieure à 10 kΩ.

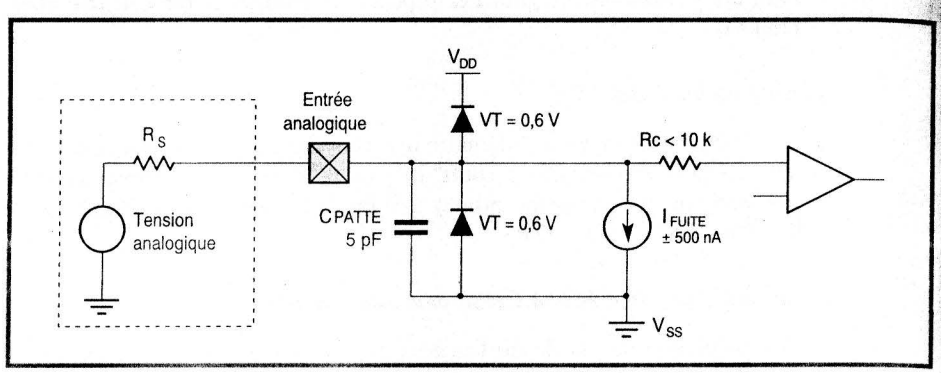


Figure 11.4 – Schéma équivalent des entrées des comparateurs analogiques.

CHAPITRE 12

LES CONVERTISSEURS ANALOGIQUES/DIGITAUX 8 ET 10 BITS

Selon la version de PIC choisie, un convertisseur analogique/digital 8 bits ou 10 bits peut faire partie des ressources internes. Ces deux modules présentant de nombreuses similitudes, nous avons décidé de vous les présenter dans le même chapitre.

12.1 Le convertisseur analogique/digital 8 bits

Ce convertisseur analogique/digital qui équipe certains PIC est un modèle à approximations successives, précédé d'un échantillonneur bloqueur et d'un multiplexeur à plusieurs entrées.

La référence de tension utilisée pour la conversion peut être définie comme étant égale à la tension d'alimentation positive V_{DD} ou à la tension appliquée sur la patte AN3/V_{REF}. Cette tension peut alors être issue d'une référence externe, ou même de la tension de référence interne si cette ressource existe aussi sur le circuit choisi.

Ce convertisseur présente en outre la particularité de pouvoir fonctionner même lorsque le microcontrôleur est en mode SLEEP.

Son synoptique interne vous est présenté *figure 12.1* et, même si cela ne se voit pas vraiment sur cette figure, trois registres sont nécessaires pour pouvoir l'utiliser : deux registres de contrôle appelés ADCON0 et ADCON1 et un registre de résultat appelé ADRES.

Les lignes d'entrée du convertisseur sont partagées avec les lignes de certains ports parallèles (généralement le port A, mais cela peut varier selon les circuits) qui doivent être configurées en entrées lorsqu'elles sont utilisées pour la conversion. Dans le cas contraire, le convertisseur lirait le niveau logique présent sur ces pattes !

Le convertisseur utilise une technique d'échantillonnage blocage qui équivaut à charger une capacité avec la tension à mesurer. De ce fait, le multiplexeur visible

© Dunod. La photocopie non autorisée est un délit.

figure 12.1 n'a pas besoin de relier en permanence l'entrée choisie à l'entrée du convertisseur proprement dit, mais seulement pendant un temps suffisant pour charger ce condensateur. Ce temps, appelé T_{AD}, dépend de la vitesse de conversion désirée et de la fréquence d'horloge du PIC.

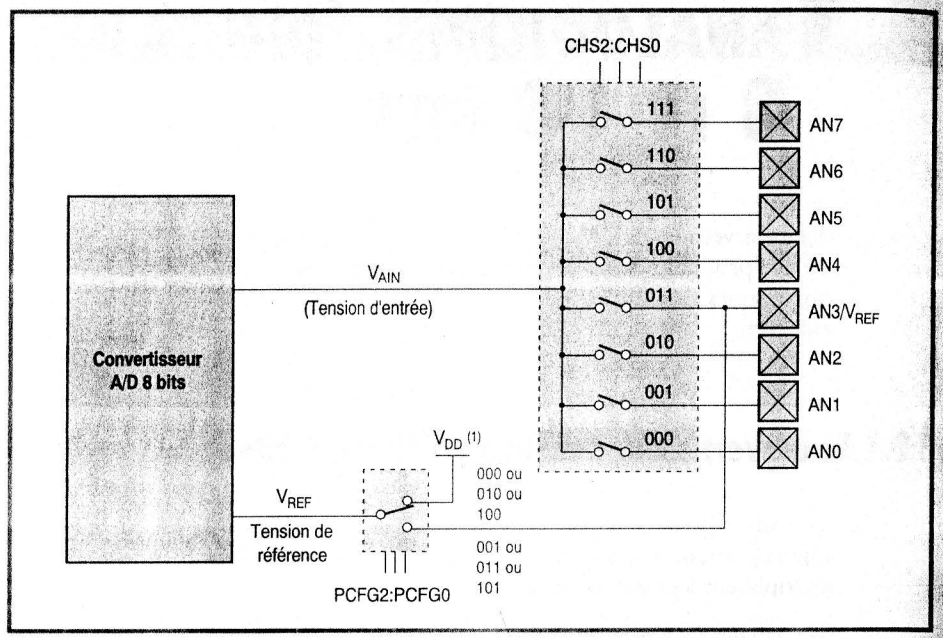


Figure 12.1 – Schéma équivalent du convertisseur analogique/digital 8 bits.

Cela étant précisé, nous pouvons étudier le contenu des registres de contrôle, ce qui nous facilitera ensuite la présentation du principe à utiliser pour programmer ce convertisseur.

Le premier registre appelé ADCON0 comporte les bits suivants :

Accès E : Écriture L : Lecture	L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E
ADCON0	**ADCS1**	**ADCS0**	**CHS2**	**CHS1**	**CHS0**	**GO/DONE**	**—**	**ADON**
État à la mise sous tension	0	0	0	0	0	0	0	0

- ***Bit 0 ou bit ADON pour AD on***

  Ce bit doit être mis à 1 pour valider le convertisseur. Dans le cas contraire, le convertisseur est arrêté et ne consomme aucune énergie.

- ***Bit 1***

  Non utilisé mais réservé pour des applications futures, ce bit doit être maintenu à 0.

- ***Bit 2 ou bit GO/DONE***

  Le fait de mettre ce bit à 1 démarre une conversion. Ensuite, tant que ce bit reste à 1 la conversion est en cours. Le passage de ce bit à 0, provoqué par la circuiterie du convertisseur, indique une fin de conversion et la possibilité de lire son résultat dans ADRES.

- ***Bits 3 à 5 ou bits CHS0 à CHS2 pour channel select***

  Ces bits permettent de choisir l'entrée à convertir, conformément aux indications du *tableau 12.1.*

- ***Bits 6 et 7 ou bits ADCS0 et ADCS1 pour AD clock select***

  Ces deux bits permettent de définir la fréquence de l'horloge de conversion, conformément aux indications du *tableau 12.2.*

Tableau 12.1 – Sélection d'entrée au moyen des bits CHS0 à CHS2 du registre ADCON0.

CHS2	CHS1	CHS0	Entrée
0	0	0	AN0
0	0	1	AN1
0	1	0	AN2
0	1	1	AN3
1	0	0	AN4
1	0	1	AN5
1	1	0	AN6
1	1	1	AN7

Tableau 12.2 – Sélection de l'horloge de conversion au moyen des bits ADCS0 et ADCS1 du registre ADCON0.

ADCS1	ADCS0	Fréquence
0	0	$F_{OSC}/2$
0	1	$F_{OSC}/8$
1	0	$F_{OSC}/32$
1	1	F_{RC} (oscillateur RC interne)

Le deuxième registre de contrôle, appelé ADCON1, ne comporte que trois bits utiles.

Accès E : Écriture L : Lecture	L	L	L	L	L	L/E	L/E	L/E
ADCON1	—	—	—	—	—	PCFG2	PCFG1	PCFG0
État à la mise sous tension	0	0	0	0	0	0	0	0

- *Bits 0 à 2 ou bits PCFG0 à PCFG2 pour port configuration*

  Ces bits permettent de choisir la répartition entre les ports analogiques et digitaux sur le ou les ports parallèles partagés avec le convertisseur, ainsi que l'utilisation comme tension de référence de V_{DD} ou de la tension appliquée sur V_{REF}. Le *tableau 12.3* résume toutes les possibilités offertes aussi bien qu'un long discours, sauf pour ce qui est du choix de la valeur de la tension de référence. Précisons donc que si AN3 est programmée pour être V_{REF}, c'est la tension appliquée sur cette entrée qui sert de tension de référence. Dans le cas contraire, c'est V_{DD} qui est utilisée comme tension de référence.

- *Bits 3 à 7*

  Inutilisés et lus comme étant à 0.

Notez aussi que, suite à un *reset*, les lignes d'entrées/sorties parallèles partagées avec celles du convertisseur analogique/digital sont toutes forcées en entrées analogiques.

© Dunod. La photocopie non autorisée est un délit.

Tableau 12.3 – Configuration des entrées du convertisseur analogique/digital 8 bits au moyen des bits PCFG0 à PCFG2 du registre ADCON1.

PCFG2	PCFG1	PCFG0	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0
0	0	0	A	A	A	A	A	A	A	A
0	0	1	A	A	A	A	V_{REF}	A	A	A
0	1	0	D	D	D	A	A	A	A	A
0	1	1	D	D	A	A	V_{REF}	A	A	A
1	0	0	D	D	D	D	A	D	A	A
1	0	1	D	D	D	D	V_{REF}	D	A	A
1	1	X	D	D	D	D	D	D	D	D

Le contenu de ces registres de contrôle étant vu, le principe d'utilisation du convertisseur est très simple à comprendre. Il suffit en effet de procéder de la façon suivante :

- configurez le convertisseur en programmant les bits des registres de contrôle ;
- validez ou non les interruptions en provenance du convertisseur en programmant le bit adéquat du registre PIEx (voir chapitre 3 si nécessaire) ;
- assurez-vous de respecter le temps minimum d'échantillonnage T_{AD} (voir ci-après) ;
- démarrez la conversion en mettant le bit GO/DONE de ADCON0 à 1 ;
- attendez la fin de conversion, soit en testant le passage à 0 de ce bit GO/DONE, soit par réception de l'interruption en provenance du convertisseur si elle a été validée ;
- le résultat de la conversion peut alors être lu dans le registre ADRES, à la suite de quoi il ne faut pas oublier de remettre le drapeau d'indication de fin de conversion ADIF du registre PIRx à 0 si nécessaire.

Il est alors possible de demander la conversion d'une autre voie (ou de la même bien sûr), mais notez qu'il est alors nécessaire d'attendre un temps au moins égal à $2\,T_{AD}$.

Notez également que, pour une raison non précisée dans la fiche technique, il ne faut pas positionner à 1 les bits ADON et GO/DONE avec la même instruction, mais procéder en deux fois.

La conversion se déroule en fait en deux phases. La première est une phase dite d'acquisition, pendant laquelle l'entrée choisie est reliée au condensateur de l'échantillonneur bloqueur. Cette phase doit durer un temps suffisant pour permettre une charge correcte de ce condensateur. Elle est suivie par la phase de conversion proprement dite, pendant laquelle la tension lue aux bornes du condensateur de l'échantillonneur bloqueur est effectivement convertie en numérique.

Comme le montre la *figure 12.2* qui présente le schéma équivalent d'une entrée, le temps de charge de ce condensateur est directement influencé par la résistance interne R_S de la source qui fournit la tension à convertir, et par la résistance interne du commutateur de l'échantillonneur R_{SS}. Cette résistance varie en fonction de la tension d'alimentation, comme indiqué sur cette même *figure 12.2*. De ce fait, le temps d'acquisition minimum nécessaire peut être déterminé par la relation :

$$T_C = -51{,}2 \text{ pF} \times (1 \text{ k}\Omega + R_{SS} + R_S) \text{ Log } (1/511)$$

avec R_{SS} et R_S exprimées en kilo-ohms et un temps donné en microsecondes (les ingénieurs de chez Microchip adorent les unités SI !).

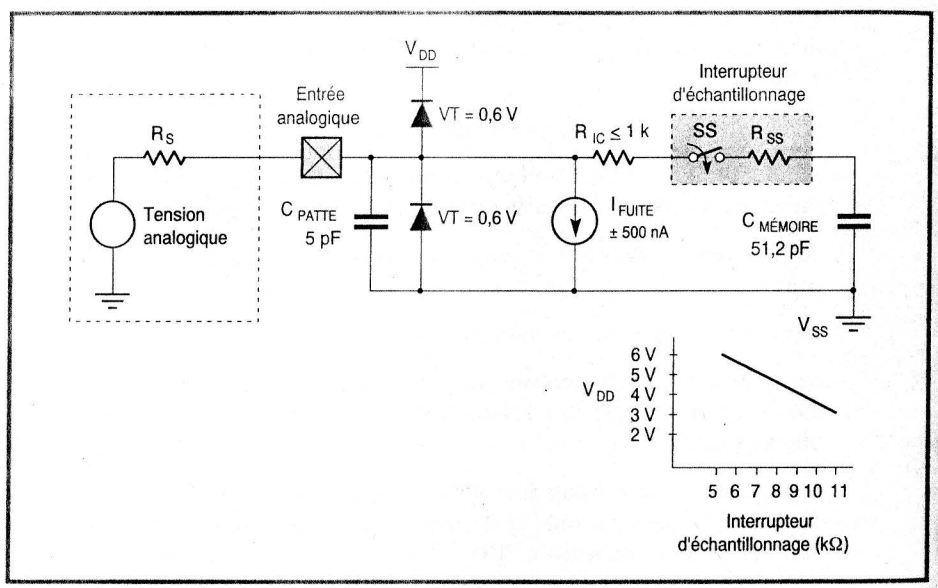


Figure 12.2 – Schéma équivalent d'une entrée analogique du convertisseur analogique/digital 8 bits.

© Dunod. La photocopie non autorisée est un délit.

L'horloge de conversion quant à elle doit être choisie sachant que la conversion nécessite un temps égal à $10\,T_{AD}$. En outre, pour assurer une conversion correcte, un temps T_{AD} minimum de 1,6 µs doit être retenu.

Les *tableaux 12.4* et *12.5* montrent quelles sont les diverses possibilités offertes par programmation des bits ADCS0 et ADCS1 du registre ADCON0 en fonction de la fréquence d'horloge du PIC choisi.

Tableau 12.4 – T_{AD} en fonction de la fréquence d'horloge pour le convertisseur analogique/digital 8 bits et les circuits en version C.

Horloge A/D		Horloge PIC			
Fréquence	ADCS1:ADCS0	20 MHz	5 MHz	1,25 MHz	333,33 kHz
$2T_{OSC}$	00	100 ns(2)	400 ns(2)	1,6 µs	6 µs
$8T_{OSC}$	01	400 ns(2)	1,6 µs	6,4 µs	24 µs(3)
$32T_{OSC}$	10	1,6 µs	6,4 µs	25,6 µs(3)	96 µs(3)
RC	11	2 à 6 µs(1,4)	2 à 6 µs(1,4)	2 à 6 µs(1,4)	2 à 6 µs(1)

(1) Le temps T_{AD} typique de l'horloge RC est de 4 µs.
(2) Ces valeurs ne respectent pas la valeur minimum de T_{AD} préconisée.
(3) La sélection d'une horloge plus rapide est conseillée pour réduire le temps de conversion.
(4) Pour une fréquence d'horloge PIC supérieure à 1 MHz, le circuit doit être en mode SLEEP pour réaliser une conversion complète et afin que la précision spécifiée soit respectée.

Tableau 12.5 – T_{AD} en fonction de la fréquence d'horloge pour le convertisseur analogique/digital 8 bits et les circuits en version LC.

Horloge A/D		Horloge PIC			
Fréquence	ADCS1:ADCS0	4 MHz	2 MHz	1,25 MHz	333,33 kHz
$2T_{OSC}$	00	500 ns(2)	1,0 µs(2)	1,6 µs(2)	6 µs
$8T_{OSC}$	01	2,0 µs(2)	4,0 µs	6,4 µs	24 µs(3)
$32T_{OSC}$	10	8,0 µs	16,0 µs	25,6 µs(3)	96 µs(3)
RC	11	3 à 9 µs(1,4)	3 à 9 µs(1,4)	3 à 9 µs(1,4)	3 à 9 µs(1)

(1) Le temps T_{AD} typique de l'horloge RC est de 6 µs.
(2) Ces valeurs ne respectent pas la valeur minimum de T_{AD} préconisée.
(3) La sélection d'une horloge plus rapide est conseillée pour réduire le temps de conversion.
(4) Pour une fréquence d'horloge PIC supérieure à 1 MHz, le circuit doit être en mode SLEEP pour réaliser une conversion complète et afin que la précision spécifiée soit respectée.

Pour une fréquence d'horloge PIC sup. à 1 MHz, le circuit doit être en mode SLEEP pour réaliser une conversion complète et afin

Notez enfin que, comme pour les comparateurs analogiques vus au chapitre précédent, il vous appartient de configurer correctement les bits du registre TRIS du port d'entrées/sorties parallèle partagé avec le convertisseur afin de placer en entrées toutes les lignes effectivement employées comme entrées analogiques.

Dans le cas contraire, celles des lignes qui seraient placées en sorties verraient leur niveau logique haut ou bas (selon le cas) converti en numérique par le convertisseur lorsque l'entrée correspondante serait sélectionnée.

12.1.1 Utilisation du convertisseur en mode SLEEP

Comme nous l'avons indiqué en début de chapitre, le fonctionnement du convertisseur est possible en mode SLEEP. Pour cela, il faut alors impérativement utiliser comme horloge de conversion le mode RC (ADCS1 et ADCS0 à 1) puisque l'horloge normale du PIC est arrêtée dans ce mode de fonctionnement.

Lorsque cette horloge est sélectionnée, le convertisseur attend un cycle instruction avant de commencer la conversion, ce qui permet d'exécuter l'instruction SLEEP.

Lorsque la conversion est terminée, son résultat est placé dans ADRES, le bit GO/DONE est mis à 0 et une interruption est générée si elle y était autorisée. Dans ce cas, le PIC est réveillé du mode SLEEP par cette dernière. Si cette interruption n'est pas autorisée, le convertisseur attend dans cet état que le processeur sorte du mode SLEEP, le résultat et le bit GO/DONE restant valables.

Lorsqu'une autre horloge que l'horloge RC est sélectionnée, le fait de passer en mode SLEEP alors qu'une conversion est en cours termine prématurément cette dernière et arrête le convertisseur afin d'économiser l'énergie.

12.1.2 Utilisation du convertisseur avec un module CCP

Nous avons vu, lors de la description du module CCP2 au chapitre 8, qu'il était possible de l'utiliser pour déclencher une conversion analogique/digitale. Cela ne change rien aux principes exposés ci-dessus ni au déroulement général de la programmation d'une conversion. La seule différence avec cet exposé tient au fait que le bit GO/DONE ordonnant le départ de la conversion n'est plus mis à 1 par vos soins, c'est-à-dire par votre programme, mais par l'intermédiaire de CCP2.

Si cette possibilité de déclenchement est utilisée par CCP2 mais que le convertisseur n'est pas validé par son bit ADON, aucune action n'a lieu au niveau du convertisseur, mais le *timer* est tout de même remis à 0 comme si l'ordre de déclenchement de la conversion avait été donné.

© Dunod. La photocopie non autorisée est un délit.

12.2 Le convertisseur analogique/digital 10 bits

Ce module présente de nombreuses similitudes avec le précédent, si ce n'est bien sûr sa précision, aussi n'allons nous voir ci-dessous que les différences avec son homologue à 8 bits.

Le schéma synoptique, quoique très proche du modèle 8 bits, s'en éloigne quelque peu au niveau de la tension de référence, comme indiqué *figure 12.3*. En effet, comme vous pouvez le constater sur ce schéma, la tension de référence n'est plus nécessairement comprise entre 0 et V_{DD} ou 0 et V_{REF} mais entre V_{REF-} et V_{REF+}.

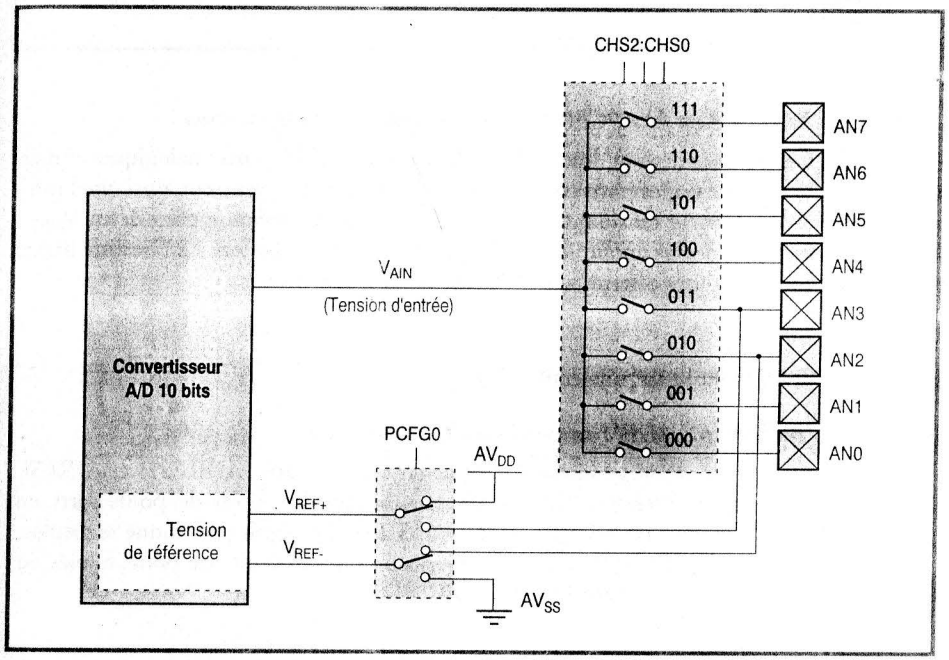


Figure 12.3 - Schéma équivalent du convertisseur analogique/digital 10 bits.

Lorsque la tension d'alimentation V_{DD} est utilisée pour V_{REF+}, V_{REF-} est nécessairement égale à V_{SS}, c'est-à-dire à une tension nulle. Par contre, lorsque l'entrée AN3 est utilisée pour V_{REF+}, la tension de référence basse peut être soit V_{SS}, soit V_{REF-}, c'est-à-dire la tension appliquée sur AN2.

La deuxième différence, qui ne se voit pas sur cette figure, se situe évidemment au niveau du registre de résultat de conversion qui est ici scindé en deux registres, 10 bits oblige, qui ont pour nom ADRESH et ADRESL.

Le registre de contrôle ADCON0 comporte les mêmes bits et fonctionne de la même façon que celui du convertisseur 8 bits, nous n'y reviendrons donc pas ici. Par contre, le registre ADCON1 est légèrement différent et voici la signification bit à bit de son contenu :

Accès E : Écriture L : Lecture	L	L	L/E	L	L/E	L/E	L/E	L/E
ADCON1 (10 bits)	—	—	ADFM	—	PCFG3	PCFG2	PCFG1	PCFG0
État à la mise sous tension	0	0	0	0	0	0	0	0

- ***Bits 0 à 3 ou bits PCFG0 à PCFG3 pour port configuration***

  Ces bits permettent de choisir la répartition entre les ports analogiques et digitaux sur le ou les ports parallèles partagés avec le convertisseur ainsi que l'utilisation comme tension de référence de V_{DD}, de la tension appliquée sur V_{REF+} ou des tensions appliquées sur V_{REF+} et V_{REF-}. Le *tableau 12.6* résume toutes les possibilités offertes aussi bien qu'un long discours.

- ***Bit 4***

  Inutilisé et lu comme étant à 0.

- ***Bit 5 ou bit ADFM pour AD result format select***

  S'il est à 1, ce bit positionne le résultat contenu dans ADRESH et ADRESL avec une justification à droite c'est-à-dire que les 6 bits de poids forts de ADRESH sont lus comme étant à 0. S'il est à 0, ce bit positionne le résultat avec une justification à gauche c'est-à-dire que les 6 bits de poids faibles de ADRESL sont lus comme étant à 0.

- ***Bits 6 et 7***

  Inutilisés et lus comme étant à 0.

Le principe de déclenchement d'une conversion est analogue à ce que nous avons vu ci-dessus pour le convertisseur 8 bits, si ce n'est que le résultat doit maintenant être lu dans deux registres : ADRESL pour les poids faibles et ADRESH pour les poids forts. La *figure 12.4* montre comment ce résultat se trouve cadré dans ces deux registres en fonction de l'état du bit ADFM vu ci-dessus.

Les contraintes de temps évoquées pour le convertisseur 8 bits se retrouvent évidemment ici aussi mais, comme le montre la *figure 12.5*, le schéma équivalent d'une entrée n'est plus tout à fait identique, car la capacité de l'échantillonneur

© Dunod. La photocopie non autorisée est un délit.

Tableau 12.6 – Configuration des entrées du convertisseur analogique/digital 10 bits au moyen des bits PCFG0 à PCFG3 du registre ADCON1.

PCFG3	PCFG2	PCFG1	PCFG0	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0	V_{REF+}	V_{REF-}
0	0	0	0	A	A	A	A	A	A	A	A	AV_{DD}	AV_{SS}
0	0	0	1	A	A	A	A	V_{REF+}	A	A	A	AN3	AV_{SS}
0	0	1	0	D	D	D	A	A	A	A	A	AV_{DD}	AV_{SS}
0	0	1	1	D	D	D	A	V_{REF+}	A	A	A	AN3	AV_{SS}
0	1	0	0	D	D	D	D	A	D	A	A	AV_{DD}	AV_{SS}
0	1	0	1	D	D	D	D	V_{REF+}	D	A	A	AN3	AV_{SS}
0	1	1	X	D	D	D	D	D	D	D	D	-	-
1	0	0	0	A	A	A	A	V_{REF+}	V_{REF-}	A	A	AN3	AN2
1	0	0	1	D	D	A	A	A	A	A	A	AV_{DD}	AV_{SS}
1	0	1	0	D	D	A	A	V_{REF+}	A	A	A	AN3	AV_{SS}
1	0	1	1	D	D	A	A	V_{REF+}	V_{REF-}	A	A	AN3	AN2
1	1	0	0	D	D	D	A	V_{REF+}	V_{REF-}	A	A	AN3	AN2
1	1	0	1	D	D	D	D	V_{REF+}	V_{REF-}	A	A	AN3	AN2
1	1	1	0	D	D	D	D	D	D	D	A	AV_{DD}	AV_{SS}
1	1	1	1	D	D	D	D	V_{REF+}	V_{REF-}	D	A	AN3	AN2

est maintenant de 120 pF. De ce fait, la relation permettant de déterminer le temps d'acquisition minimum T_C devient :

$$T_C = -120 \text{ pF} \times (1 \text{ k}\Omega + R_{SS} + R_S) \text{ Log } (1/2\,047)$$

avec R_{SS} et R_S exprimées en kilo-ohms et un temps donné en microsecondes.

L'horloge de conversion quant à elle doit être choisie sachant que la conversion nécessite maintenant un temps égal à $11{,}5T_{AD}$. En outre, pour assurer une conversion correcte, un temps T_{AD} minimum de 1,6 µs doit toujours être retenu.

Les *tableaux 12.7* et *12.8* montrent quelles sont les diverses possibilités offertes par programmation des bits ADCS0 et ADCS1 du registre ADCON0 en fonction de la fréquence d'horloge du PIC choisi.

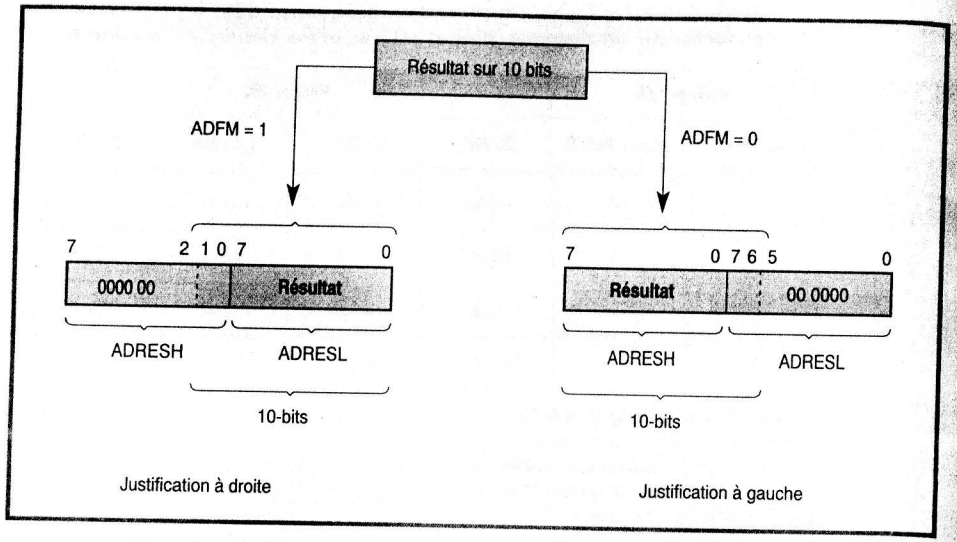


Figure 12.4 – Les deux modes de cadrage du résultat dans les registres ADRESH et ADRESL du convertisseur analogique/digital 10 bits.

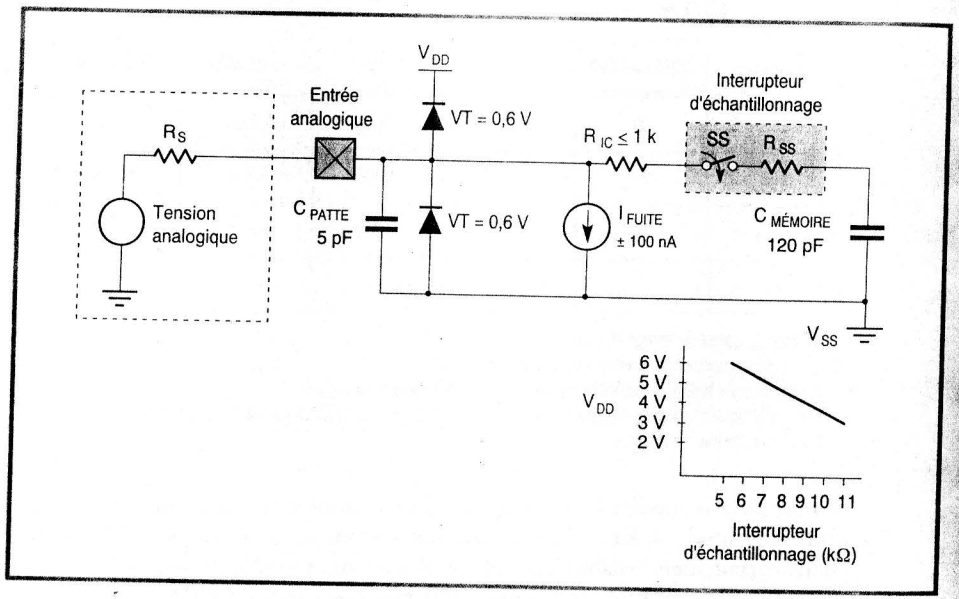


Figure 12.5 – Schéma équivalent d'une entrée analogique du convertisseur analogique/digital 10 bits.

© Dunod. La photocopie non autorisée est un délit.

Tableau 12.7 – T_{AD} en fonction de la fréquence d'horloge pour le convertisseur analogique/digital 10 bits et les circuits en version C.

Horloge A/D		Horloge PIC			
Fréquence	ADCS1:ADCS0	20 MHz	5 MHz	1,25 MHz	333,33 kHz
$2T_{OSC}$	00	100 ns(2)	400 ns(2)	1,6 µs	6 µs
$8T_{OSC}$	01	400 ns(2)	1,6 µs	6,4 µs	24 µs(3)
$32T_{OSC}$	10	1,6 µs	6,4 µs	25,6 µs(3)	96 µs(3)
RC	11	2 à 6 µs(1,4)	2 à 6 µs(1,4)	2 à 6 µs(1,4)	2 à 6 µs(1)

(1) Le temps T_{AD} typique de l'horloge RC est de 4 µs.
(2) Ces valeurs ne respectent pas la valeur minimum de T_{AD} préconisée.
(3) La sélection d'une horloge plus rapide est conseillée pour réduire le temps de conversion.
(4) Pour une fréquence d'horloge PIC supérieure à 1 MHz, le circuit doit être en mode SLEEP pour réaliser une conversion complète et afin que la précision spécifiée soit respectée.

Tableau 12.8 – T_{AD} en fonction de la fréquence d'horloge pour le convertisseur analogique/digital 10 bits et les circuits en version LC.

Horloge A/D		Horloge PIC			
Fréquence	ADCS1:ADCS0	4 MHz	2 MHz	1,25 MHz	333,33 kHz
$2T_{OSC}$	00	500 ns(2)	1,0 µs(2)	1,6 µs(2)	6 µs
$8T_{OSC}$	01	2,0 µs(2)	4,0 µs	6,4 µs	24 µs(3)
$32T_{OSC}$	10	8,0 µs	16,0 µs	25,6 µs(3)	96 µs(3)
RC	11	3 à 9 µs(1,4)	3 à 9 µs(1,4)	3 à 9 µs(1,4)	3 à 9 µs(1)

(1) Le temps T_{AD} typique de l'horloge RC est de 6 µs.
(2) Ces valeurs ne respectent pas la valeur minimum de T_{AD} préconisée.
(3) La sélection d'une horloge plus rapide est conseillée pour réduire le temps de conversion.
(4) Pour une fréquence d'horloge PIC supérieure à 1 MHz, le circuit doit être en mode SLEEP pour réaliser une conversion complète et afin que la précision spécifiée soit respectée.

Comme pour le modèle 8 bits, il est possible ici aussi de réaliser une conversion pendant le mode SLEEP, tout ce que nous avons écrit ci-dessus à ce propos restant parfaitement valable. Par contre, il n'est pas possible de déclencher une conversion au moyen du module de comparaison et capture CCP2.

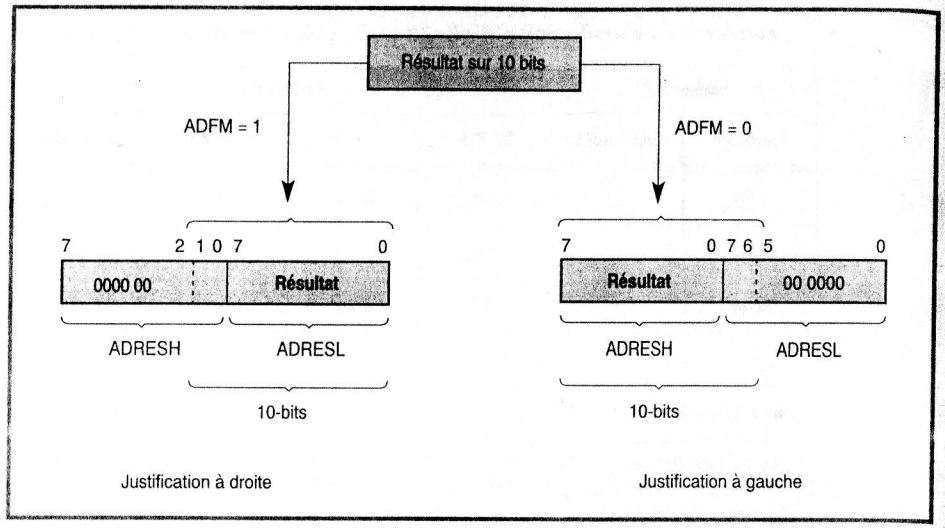


Figure 12.4 – Les deux modes de cadrage du résultat dans les registres ADRESH et ADRESL du convertisseur analogique/digital 10 bits.

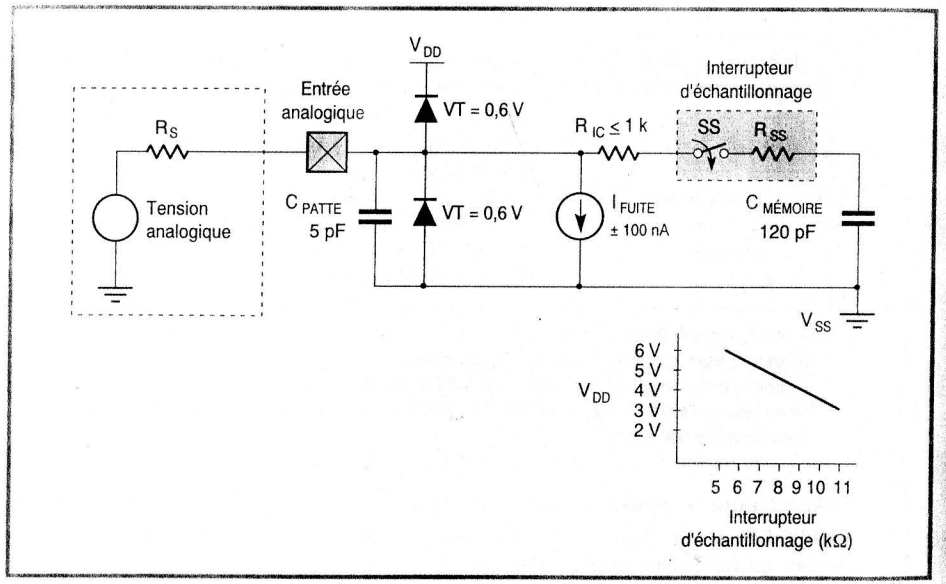


Figure 12.5 – Schéma équivalent d'une entrée analogique du convertisseur analogique/digital 10 bits.

© Dunod. La photocopie non autorisée est un délit.

Tableau 12.7 – T_{AD} en fonction de la fréquence d'horloge pour le convertisseur analogique/digital 10 bits et les circuits en version C.

Horloge A/D		Horloge PIC			
Fréquence	ADCS1:ADCS0	20 MHz	5 MHz	1,25 MHz	333,33 kHz
$2T_{OSC}$	00	100 ns(2)	400 ns(2)	1,6 µs	6 µs
$8T_{OSC}$	01	400 ns(2)	1,6 µs	6,4 µs	24 µs(3)
$32T_{OSC}$	10	1,6 µs	6,4 µs	25,6 µs(3)	96 µs(3)
RC	11	2 à 6 µs(1,4)	2 à 6 µs(1,4)	2 à 6 µs(1,4)	2 à 6 µs(1)

(1) Le temps T_{AD} typique de l'horloge RC est de 4 µs.
(2) Ces valeurs ne respectent pas la valeur minimum de T_{AD} préconisée.
(3) La sélection d'une horloge plus rapide est conseillée pour réduire le temps de conversion.
(4) Pour une fréquence d'horloge PIC supérieure à 1 MHz, le circuit doit être en mode SLEEP pour réaliser une conversion complète et afin que la précision spécifiée soit respectée.

Tableau 12.8 – T_{AD} en fonction de la fréquence d'horloge pour le convertisseur analogique/digital 10 bits et les circuits en version LC.

Horloge A/D		Horloge PIC			
Fréquence	ADCS1:ADCS0	4 MHz	2 MHz	1,25 MHz	333,33 kHz
$2T_{OSC}$	00	500 ns(2)	1,0 µs(2)	1,6 µs(2)	6 µs
$8T_{OSC}$	01	2,0 µs(2)	4,0 µs	6,4 µs	24 µs(3)
$32T_{OSC}$	10	8,0 µs	16,0 µs	25,6 µs(3)	96 µs(3)
RC	11	3 à 9 µs(1,4)	3 à 9 µs(1,4)	3 à 9 µs(1,4)	3 à 9 µs(1)

(1) Le temps T_{AD} typique de l'horloge RC est de 6 µs.
(2) Ces valeurs ne respectent pas la valeur minimum de T_{AD} préconisée.
(3) La sélection d'une horloge plus rapide est conseillée pour réduire le temps de conversion.
(4) Pour une fréquence d'horloge PIC supérieure à 1 MHz, le circuit doit être en mode SLEEP pour réaliser une conversion complète et afin que la précision spécifiée soit respectée.

Comme pour le modèle 8 bits, il est possible ici aussi de réaliser une conversion pendant le mode SLEEP, tout ce que nous avons écrit ci-dessus à ce propos restant parfaitement valable. Par contre, il n'est pas possible de déclencher une conversion au moyen du module de comparaison et capture CCP2.

CHAPITRE 13

L'INTERFACE POUR AFFICHEURS À CRISTAUX LIQUIDES

Certains PIC récents disposent d'origine d'un port, ou plus exactement d'une interface, pour afficheurs à cristaux liquides (LCD). Cette interface ne doit pas être confondue avec les sorties pour afficheurs à cristaux liquides présentes sur certains circuits intégrés. En effet, elle n'est pas limitée à la seule commande d'afficheurs numériques ni même alphanumériques standards, mais permet réellement de piloter n'importe quel type d'afficheur LCD comportant de 1 à 128 segments distincts, ces derniers pouvant fonctionner, selon leur nombre, en mode statique ou selon divers types de multiplexages.

13.1 Principe général

La *figure 13.1* présente un schéma synoptique simplifié de cette interface qui va nous permettre de comprendre comment elle fonctionne.

L'interface peut commander directement jusqu'à 32 segments, qui peuvent alors fonctionner en mode statique s'ils sont seuls, en mode multiplexé avec 2 communs si leur nombre est compris entre 32 et 64, et ainsi de suite jusqu'à un multiplexage avec 4 communs si leur nombre est compris entre 96 et 128.

Même si cela ne se voit pas intégralement sur cette figure, l'interface est commandée par dix-neuf registres. Trois registres, appelés LCDCON, LCDPS et LCDSE, servent à configurer l'interface alors que les seize autres, appelés LCDD00 à LCDD15, contiennent les données servant à coder l'état des segments.

Un afficheur à cristaux liquides nécessitant impérativement une horloge pour fonctionner, celle-ci est également fournie par l'interface à partir d'une des trois sources visibles en bas de la *figure 13.1*, à savoir : l'horloge système interne, une horloge provenant d'un oscillateur RC interne autonome ou encore une horloge fournie depuis l'extérieur via la patte T1CKI du *timer 1*.

© Dunod. La photocopie non autorisée est un délit.

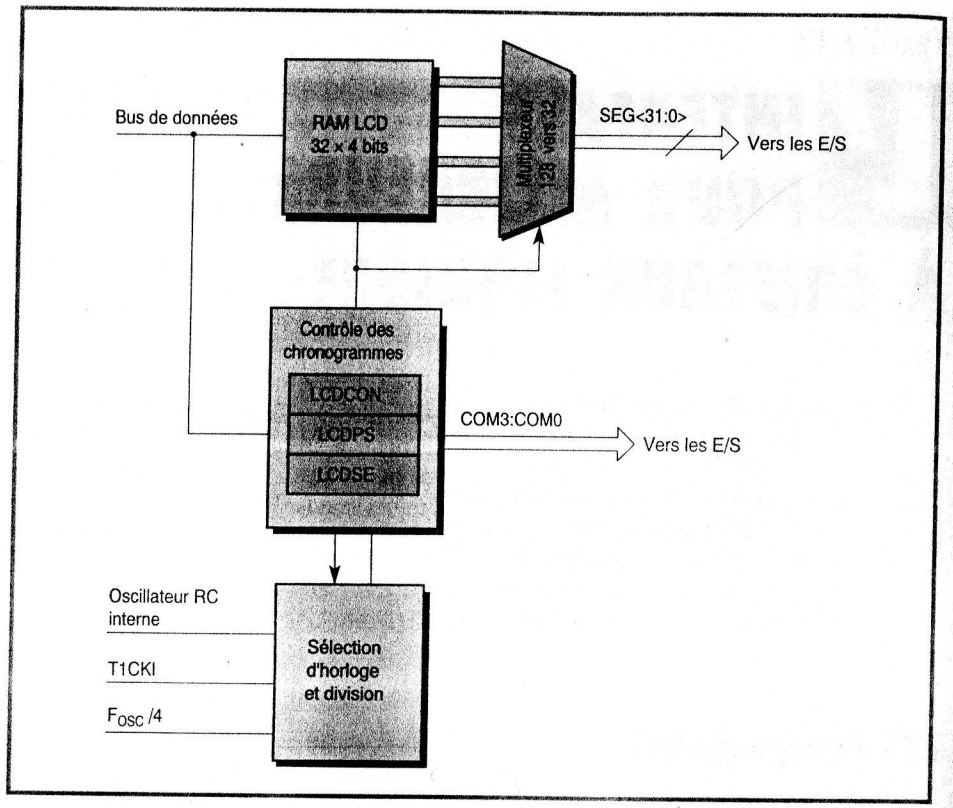


Figure 13.1 – Schéma équivalent de l'interface pour afficheurs LCD.

Le principe général d'utilisation de cette interface est relativement simple pour qui est un peu habitué aux afficheurs à cristaux liquides. En effet, si leur fonctionnement en mode statique ne pose généralement pas de problème, leur utilisation en mode multiplexé est un peu plus délicate, mais l'interface LCD des PIC n'y est pour rien…

Pour piloter un afficheur LCD, il suffit donc d'initialiser les trois registres de contrôle en fonction des caractéristiques de l'afficheur (nombre de segments, nombre de communs, type de multiplexage, fréquence de trame, etc.). La commande de l'affichage proprement dite consiste ensuite tout simplement à positionner à 0 ou à 1 les données dans les registres de segments LCDD00 à LCDD15 en fonction de ceux des segments qui doivent être éteints ou allumés.

Au moyen du seul bit LCDEN du registre LCDCON, il est également possible d'éteindre ou d'allumer tout l'afficheur, afficheur que vous pouvez décider ou non de faire fonctionner lorsque le microcontrôleur est en mode SLEEP.

© Dunod. La photocopie non autorisée est un délit.

Le registre de contrôle principal est LCDCON, dont voici le contenu et la fonction bit par bit :

Accès E : Écriture L : Lecture	L/E	L/E	L	L/E	L/E	L/E	L/E	L/E
LCDCON	LCDEN	SLPEN	—	VGEN	CS1	CS0	LMUX1	LMUX0
État à la mise sous tension	0	0	0	0	0	0	0	0

- *Bits 0 et 1 ou bits LMUX0 et LMUX1*

  Ces deux bits permettent de préciser le nombre d'électrodes de commande de segments et d'électrodes communes, ainsi que le mode de multiplexage puisque ces deux informations sont liées, conformément aux indications du *tableau 13.1.*

- *Bits 2 et 3 ou bits CS0 et CS1 pour clock source*

  Ces deux bits permettent de sélectionner la source d'horloge de multiplexage, conformément aux indications du *tableau 13.2.*

- *Bit 4 ou bit VGEN pour voltage generator enable*

  Lorsqu'il est mis à 1, ce bit valide le générateur de tension d'alimentation interne à destination de l'afficheur. S'il est mis à 0, ce générateur est arrêté et la tension d'alimentation de l'afficheur doit provenir d'une source externe.

- *Bit 5*

  Inutilisé et lu comme étant à 0.

- *Bit 6 ou bit SLPEN pour sleep enable*

  Lorsqu'il est mis à 1, ce bit arrête l'affichage lorsque le PIC passe en mode SLEEP. S'il est mis à 0, l'affichage continue à fonctionner même lorsque le circuit est en mode SLEEP.

- *Bit 7 ou bit LCDEN pour lcd enable*

  Lorsqu'il est mis à 1, ce bit valide l'interface LCD. Elle est arrêtée dans le cas contraire.

Le registre LCDPS permet quant à lui de sélectionner la fréquence de trame effectivement utilisée pour piloter l'affichage au moyen de ses quatre bits significatifs présentés page suivante :

Tableau 13.1 – Sélection du mode de fonctionnement au moyen des bits LMUX0 et LMUX1 du registre LCDCON.

LMUX1	LMUX0	Multiplex	Polarisation	Nombre maximal de segments
0	0	Statique (COM0)	Statique	32
0	1	1/2 (COM0, COM1)	1/3	31
1	0	1/3 (COM0, COM1, COM2)	1/3	30
1	1	1/4 (COM0, COM1, COM2, COM3)	1/3	29

Tableau 13.2 – Sélection de la source d'horloge au moyen des bits CS0 et CS1 du registre LCDCON.

CS1	CS0	Horloge
0	0	$F_{OSC}/256$
0	1	T1CKI (*timer 1*)
1	X	Oscillateur RC interne

Accès E : Écriture L : Lecture	L	L	L	L	L/E	L/E	L/E	L/E
LCDPS	—	—	—	—	LP3	LP2	LP1	LP0
État à la mise sous tension	0	0	0	0	X	X	X	X

- *Bits 0 à 3 ou bits LP0 à LP3*

  Ces bits servent à définir la fréquence de l'horloge de trame de l'afficheur conformément aux indications du *tableau 13.3*.

- *Bits 4 à 7*

  Inutilisés et lus comme étant à 0.

Enfin, le dernier registre de contrôle appelé LCDSE permet de définir quelles sont les lignes du port LCD qui seront effectivement utilisées pour piloter un afficheur de ce type et quelle sont celles qui pourront servir de lignes de port parallèle à usage général, mais attention, dans ce cas la majorité d'entre elles ne peut fonctionner qu'en entrée.

© Dunod. La photocopie non autorisée est un délit.

Tableau 13.3 – Sélection de la fréquence de trame au moyen des bits LP0 à LP3 du registre LCDPS.

LMUX1	LMUX0	Multiplex	Fréquence de trame
0	0	Statique	Horloge/(128 · (LP3 : LP0 + 1))
0	1	1/2	Horloge/(128 · (LP3 : LP0 + 1))
1	0	1/3	Horloge/(96 · (LP3 : LP0 + 1))
1	1	1/4	Horloge/(128 · (LP3 : LP0 + 1))

Accès E : Écriture L : Lecture	L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E
LCDSE	SE29	SE27	SE20	SE16	SE12	SE9	SE5	SE0
État à la mise sous tension	1	1	1	1	1	1	1	1

- *Bit 0 ou bit SE0 pour segment 0*

  Ce bit contrôle les lignes SEG00 à SEG04. Lorsqu'il est à 1, elles sont effectivement utilisées pour piloter un afficheur LCD. Elles servent de port d'entrée/sortie parallèle dans le cas contraire.

- *Bit 1 ou bit SE5 pour segment 5*

  Ce bit contrôle les lignes SEG05 à SEG08. Lorsqu'il est à 1, elles sont effectivement utilisées pour piloter un afficheur LCD. Elles servent de port d'entrée parallèle dans le cas contraire.

- *Bit 2 ou bit SE9 pour segment 9*

  Ce bit contrôle les lignes SEG09 à SEG11. Lorsqu'il est à 1, elles sont effectivement utilisées pour piloter un afficheur LCD. Elles servent de port d'entrée parallèle dans le cas contraire.

- *Bit 3 ou bit SE12 pour segment 12*

  Ce bit contrôle les lignes SEG12 à SEG15. Lorsqu'il est à 1, elles sont effectivement utilisées pour piloter un afficheur LCD. Elles servent de port d'entrée parallèle dans le cas contraire.

Tableau 13.1 – Sélection du mode de fonctionnement au moyen des bits LMUX0 et LMUX1 du registre LCDCON.

LMUX1	LMUX0	Multiplex	Polarisation	Nombre maximal de segments
0	0	Statique (COM0)	Statique	32
0	1	1/2 (COM0, COM1)	1/3	31
1	0	1/3 (COM0, COM1, COM2)	1/3	30
1	1	1/4 (COM0, COM1, COM2, COM3)	1/3	29

Tableau 13.2 – Sélection de la source d'horloge au moyen des bits CS0 et CS1 du registre LCDCON.

CS1	CS0	Horloge
0	0	$F_{OSC}/256$
0	1	T1CKI (*timer 1*)
1	X	Oscillateur RC interne

Accès E : Écriture L : Lecture	L	L	L	L	L/E	L/E	L/E	L/E
LCDPS	—	—	—	—	LP3	LP2	LP1	LP0
État à la mise sous tension	0	0	0	0	X	X	X	X

- *Bits 0 à 3 ou bits LP0 à LP3*

  Ces bits servent à définir la fréquence de l'horloge de trame de l'afficheur conformément aux indications du *tableau 13.3*.

- *Bits 4 à 7*

  Inutilisés et lus comme étant à 0.

Enfin, le dernier registre de contrôle appelé LCDSE permet de définir quelles sont les lignes du port LCD qui seront effectivement utilisées pour piloter un afficheur de ce type et quelle sont celles qui pourront servir de lignes de port parallèle à usage général, mais attention, dans ce cas la majorité d'entre elles ne peut fonctionner qu'en entrée.

© Dunod. La photocopie non autorisée est un délit.

Tableau 13.3 – Sélection de la fréquence de trame au moyen des bits LP0 à LP3 du registre LCDPS.

LMUX1	LMUX0	Multiplex	Fréquence de trame
0	0	Statique	Horloge/(128 · (LP3 : LP0 + 1))
0	1	1/2	Horloge/(128 · (LP3 : LP0 + 1))
1	0	1/3	Horloge/(96 · (LP3 : LP0 + 1))
1	1	1/4	Horloge/(128 · (LP3 : LP0 + 1))

Accès E : Écriture L : Lecture	L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E
LCDSE	SE29	SE27	SE20	SE16	SE12	SE9	SE5	SE0
État à la mise sous tension	1	1	1	1	1	1	1	1

- *Bit 0 ou bit SE0 pour segment 0*

  Ce bit contrôle les lignes SEG00 à SEG04. Lorsqu'il est à 1, elles sont effectivement utilisées pour piloter un afficheur LCD. Elles servent de port d'entrée/sortie parallèle dans le cas contraire.

- *Bit 1 ou bit SE5 pour segment 5*

  Ce bit contrôle les lignes SEG05 à SEG08. Lorsqu'il est à 1, elles sont effectivement utilisées pour piloter un afficheur LCD. Elles servent de port d'entrée parallèle dans le cas contraire.

- *Bit 2 ou bit SE9 pour segment 9*

  Ce bit contrôle les lignes SEG09 à SEG11. Lorsqu'il est à 1, elles sont effectivement utilisées pour piloter un afficheur LCD. Elles servent de port d'entrée parallèle dans le cas contraire.

- *Bit 3 ou bit SE12 pour segment 12*

  Ce bit contrôle les lignes SEG12 à SEG15. Lorsqu'il est à 1, elles sont effectivement utilisées pour piloter un afficheur LCD. Elles servent de port d'entrée parallèle dans le cas contraire.

- *Bit 4 ou bit SE16 pour segment 16*

  Ce bit contrôle les lignes SEG16 à SEG19. Lorsqu'il est à 1, elles sont effectivement utilisées pour piloter un afficheur LCD. Elles servent de port d'entrée parallèle dans le cas contraire.

- *Bit 5 ou bit SE20 pour segment 20*

  Ce bit contrôle les lignes SEG20 à SEG26. Lorsqu'il est à 1, elles sont effectivement utilisées pour piloter un afficheur LCD. Elles servent de port d'entrée parallèle dans le cas contraire.

- *Bit 6 ou bit SE27 pour segment 27*

  Ce bit contrôle les lignes SEG27 et SEG28. Lorsqu'il est à 1, elles sont effectivement utilisées pour piloter un afficheur LCD. Elles servent de port d'entrée parallèle dans le cas contraire.

- *Bit 7 ou bit SE29 pour segment 29*

  Ce bit contrôle les lignes COM3/SEG29 à COM1/SEG31. Lorsqu'il est à 1, elles sont effectivement utilisées pour piloter un afficheur LCD. Elles servent de port d'entrée parallèle dans le cas contraire. Ce bit est toutefois de priorité inférieure à celle des bits LMUX0 et LMUX1 du registre LCDCON qui « réquisitionnent » les lignes SEG29 à SEG31, si nécessaire, en tant qu'électrodes communes conformément au type de multiplexage choisi.

Les autres registres utilisés pour commander l'interface sont les registres « de données » qui ont pour noms LCDD00 à LCDD15. Ils permettent d'indiquer à chaque instant quel segment doit être allumé (bit à 1) ou éteint (bit à 0). Les correspondances entre les bits de ces registres et les coordonnées de chaque segment, données sous la forme : n° de segment - n° d'électrode commune, sont indiquées *tableau 13.4*.

13.2 Horloges et multiplexage

Comme nous l'avons vu lors de la présentation du registre LCDCON, trois sources d'horloge sont à votre disposition pour piloter l'horloge de multiplexage. La *figure 13.2* présente, de façon synoptique, ces trois possibilités et les divers bits agissant sur l'horloge finale produite.

L'horloge RC interne est un module totalement autonome qui continue à fonctionner même lorsque le PIC est en mode SLEEP. Cette horloge fonctionne à la vitesse nominale de 44 kHz et s'arrête lorsqu'elle n'est pas sélectionnée comme source pour l'interface LCD ou lorsque l'interface LCD n'est pas validée.

© Dunod. La photocopie non autorisée est un délit.

Tableau 13.4 – Correspondance entre les coordonnées des segments et les bits des registres LCCD00 à LCDD15.

Nom	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
LCDD00	SEG07 COM0	SEG06 COM0	SEG05 COM0	SEG04 COM0	SEG03 COM0	SEG02 COM0	SEG01 COM0	SEG00 COM0
LCDD01	SEG15 COM0	SEG14 COM0	SEG13 COM0	SEG12 COM0	SEG11 COM0	SEG10 COM0	SEG09 COM0	SEG08 COM0
LCDD02	SEG23 COM0	SEG22 COM0	SEG21 COM0	SEG20 COM0	SEG19 COM0	SEG18 COM0	SEG17 COM0	SEG16 COM0
LCDD03	SEG31 COM0	SEG30 COM0	SEG29 COM0	SEG28 COM0	SEG27 COM0	SEG26 COM0	SEG25 COM0	SEG24 COM0
LCDD04	SEG07 COM1	SEG06 COM1	SEG05 COM1	SEG04 COM1	SEG03 COM1	SEG02 COM1	SEG01 COM1	SEG00 COM1
LCDD05	SEG15 COM1	SEG14 COM1	SEG13 COM1	SEG12 COM1	SEG11 COM1	SEG10 COM1	SEG09 COM1	SEG08 COM1
LCDD06	SEG23 COM1	SEG22 COM1	SEG21 COM1	SEG20 COM1	SEG19 COM1	SEG18 COM1	SEG17 COM1	SEG16 COM1
LCDD07	SEG31 COM1[1]	SEG30 COM1	SEG29 COM1	SEG28 COM1	SEG27 COM1	SEG26 COM1	SEG25 COM1	SEG24 COM1
LCDD08	SEG07 COM2	SEG06 COM2	SEG05 COM2	SEG04 COM2	SEG03 COM2	SEG02 COM2	SEG01 COM2	SEG00 COM2
LCDD09	SEG15 COM2	SEG14 COM2	SEG13 COM2	SEG12 COM2	SEG11 COM2	SEG10 COM2	SEG09 COM2	SEG08 COM2
LCDD10	SEG23 COM2	SEG22 COM2	SEG21 COM2	SEG20 COM2	SEG19 COM2	SEG18 COM2	SEG17 COM2	SEG16 COM2
LCDD11	SEG31 COM2[1]	SEG30 COM2[1]	SEG29 COM2	SEG28 COM2	SEG27 COM2	SEG26 COM2	SEG25 COM2	SEG24 COM2
LCDD12	SEG07 COM3	SEG06 COM3	SEG05 COM3	SEG04 COM3	SEG03 COM3	SEG02 COM3	SEG01 COM3	SEG00 COM3
LCDD13	SEG15 COM3	SEG14 COM3	SEG13 COM3	SEG12 COM3	SEG11 COM3	SEG10 COM3	SEG09 COM3	SEG08 COM3
LCDD14	SEG23 COM3	SEG22 COM3	SEG21 COM3	SEG20 COM3	SEG19 COM3	SEG18 COM3	SEG17 COM3	SEG16 COM3
LCDD15	SEG31 COM3[1]	SEG30 COM3[1]	SEG29 COM3[1]	SEG28 COM3	SEG27 COM3	SEG26 COM3	SEG25 COM3	SEG24 COM3

[1] Ces segments ne sont pas affichés mais leurs emplacements peuvent être utilisés comme de la RAM à usage général.

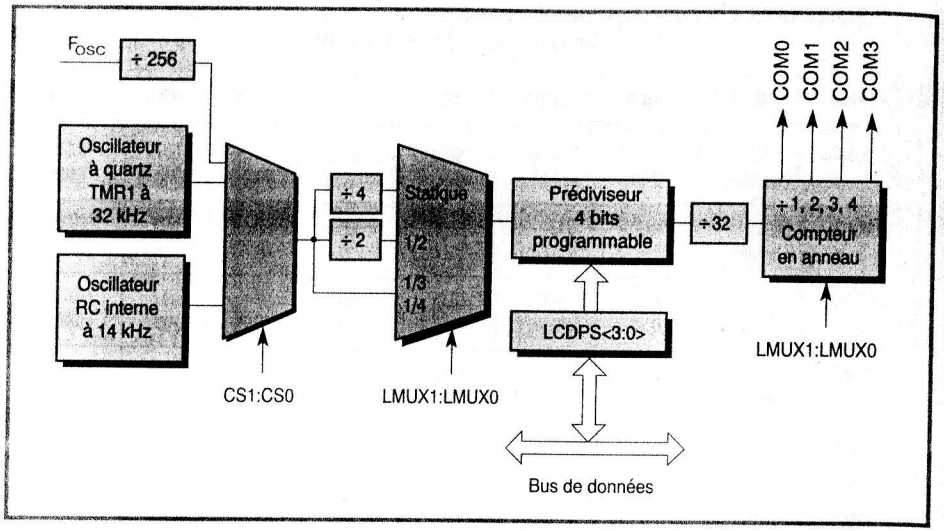


Figure 13.2 – Schéma équivalent de la circuiterie de génération d'horloge de l'interface pour afficheurs LCD.

L'horloge externe provenant du *timer 1* est la deuxième source possible. Elle utilise l'oscillateur à quartz externe proposé par le *timer 1* (voir chapitre 7 si nécessaire) mais est prévue pour une vitesse de fonctionnement typique de 32 kHz. Pour l'utiliser, il suffit juste de valider ce mode de fonctionnement au niveau du registre de contrôle T1CON du *timer 1* en positionnant son bit 3 à 1. Comme l'horloge RC interne, cette horloge continue à fonctionner même lorsque le processeur est en mode sommeil.

La troisième source d'horloge est l'horloge système divisée par 256. Ce taux de division fixe interne a été choisi de façon à disposer d'une horloge « LCD » de 32 kHz lorsque l'on travaille avec un quartz à 8 MHz.

Les *tableaux 13.5* et *13.6* indiquent les fréquences de trame obtenues au niveau de l'afficheur en fonction de la source d'horloge choisie, pour les divers modes de multiplexage utilisés et en fonction de la programmation des bits LP0 à LP3 du registre LCDPS.

Les différentes formes d'onde produites par l'interface LCD dépendent évidemment du type de multiplexage utilisé. Les *figures 13.3* à *13.6* (p. 172 à 175) présentent les quatre types de chronogrammes que l'on peut rencontrer. Les différents niveaux de tensions visibles sur ces chronogrammes sont générés en interne ou à partir d'une source externe, comme nous allons le voir dans un instant.

L'interface LCD est capable de générer une interruption, pour peu qu'elle y ait été autorisée par le bit adéquat du registre PIEx (voir chapitre 3 si nécessaire). Cette

© Dunod. La photocopie non autorisée est un délit.

interruption sert à signaler le début d'une nouvelle trame d'affichage, ce qui permet au microcontrôleur d'écrire les nouvelles données à afficher à cet instant. En procédant de la sorte, on assure une transition précise et nette d'une donnée affichée à une autre, alors qu'une écriture à n'importe quel instant de la trame provoque un léger effet de flou.

Tableau 13.5 – Fréquence de trame approximative en utilisant le *timer 1* à 32,768 kHz.

LP3 : LP0	Statique	1/2	1/3	1/4
2	85	85	114	85
3	64	64	85	64
4	51	51	68	51
5	43	43	57	43
6	37	37	49	37
7	32	32	43	32

Tableau 13.6 – Fréquence de trame approximative en utilisant l'oscillateur RC interne à 14 kHz.

LP3 : LP0	Statique	1/2	1/3	1/4
0	109	109	146	109
1	55	55	73	55
2	36	36	49	36
3	27	27	36	27

13.3 Génération des tensions de l'afficheur

La génération des différents niveaux de tension, utilisés par l'afficheur, que vous avez pu découvrir sur les chronogrammes des *figures 13.3* à *13.6*, peut être réalisée de deux manières différentes : avec un réseau de résistances en échelle si la tension d'alimentation V_{DD} disponible est suffisante, ou avec un système à pompe de charge dans le cas contraire.

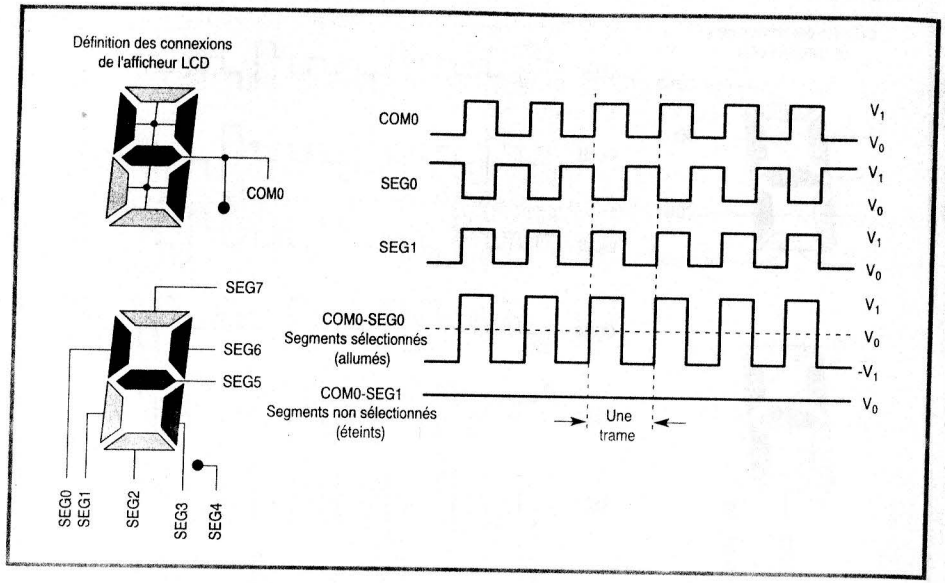


Figure 13.3 – Chronogrammes en mode statique.

Dans les deux cas, l'interface LCD intégrée fournit la majorité des ressources nécessaires et seuls quelques composants passifs, résistances et/ou condensateurs, externes sont nécessaires. La *figure 13.7* présente le synoptique de la partie de cette interface utilisée pour la génération de ces tensions, ainsi que les divers modes de connexion utilisables.

L'utilisation en mode pompe de charge (bit VGEN de LCDCON à 1) fait appel à un générateur de tension interne très stable, du type band-gap, produisant une tension comprise entre 1 et 2,3 V. Cette tension est ajustable au moyen d'un potentiomètre connecté entre la patte VLCDADJ et la masse, comme indiqué en haut de la *figure 13.7*. Ce potentiomètre sert donc en fait à régler le contraste de l'afficheur, et la tension de base ainsi choisie est disponible sur V_{LCD1}. La pompe de charge produit ensuite sur V_{LCD2} une tension double et sur V_{LCD3} une tension triple de cette valeur V_{LCD1}.

Lorsque la pompe de charge n'est pas utilisée (bit VGEN de LCDCON à 0), c'est-à-dire que l'on fait appel à l'un des deux schémas du bas de la *figure 13.7*, la sortie V_{LCD3} est reliée en interne à V_{DD}, comme le laisse entendre cette même figure.

© Dunod. La photocopie non autorisée est un délit.

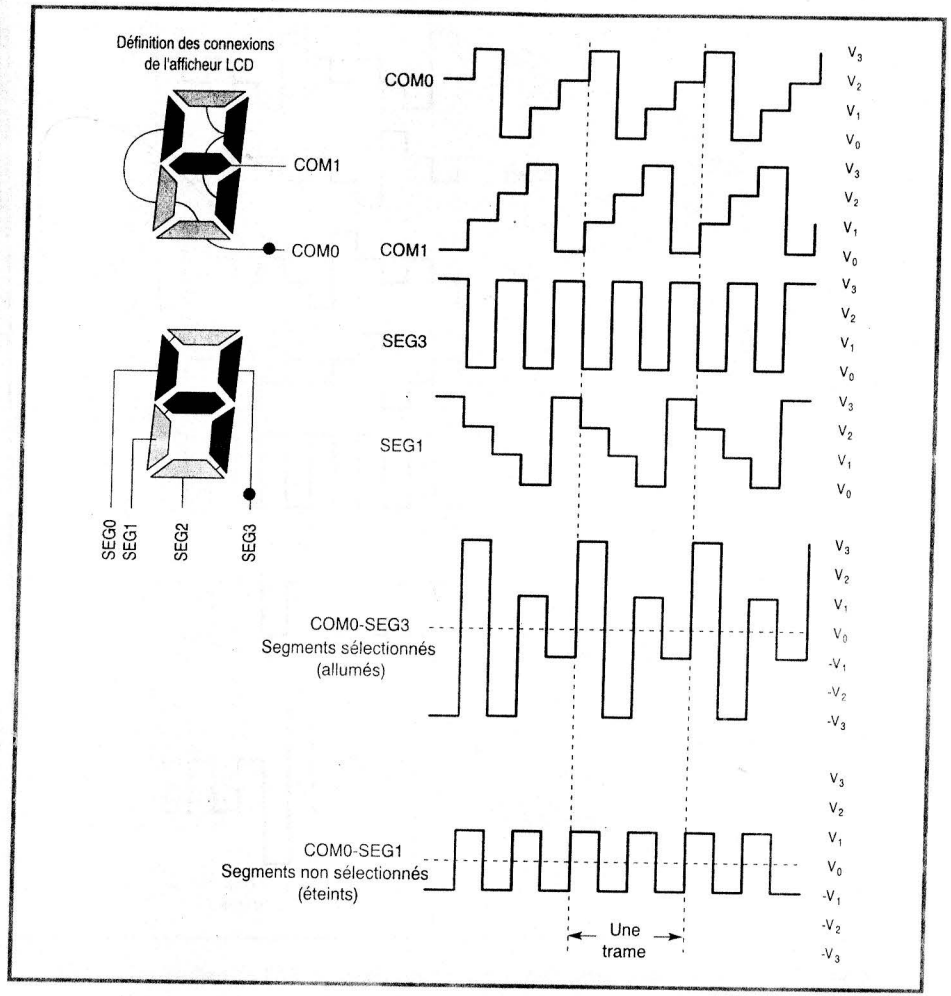


Figure 13.4 – Chronogrammes en mode multiplexage 1/2 et polarisation 1/3.

13.4 Utilisation de l'interface LCD

Suite à un *reset*, l'interface LCD est désactivée mais toutes ses lignes sont configurées de manière à piloter un afficheur à cristaux liquides et non en ports d'entrées/sorties à usage général. Cela permet d'être certain que le microcontrôleur ne pourra pas endommager l'afficheur au moyen de tensions continues permanentes appliquées à un ou plusieurs segments.

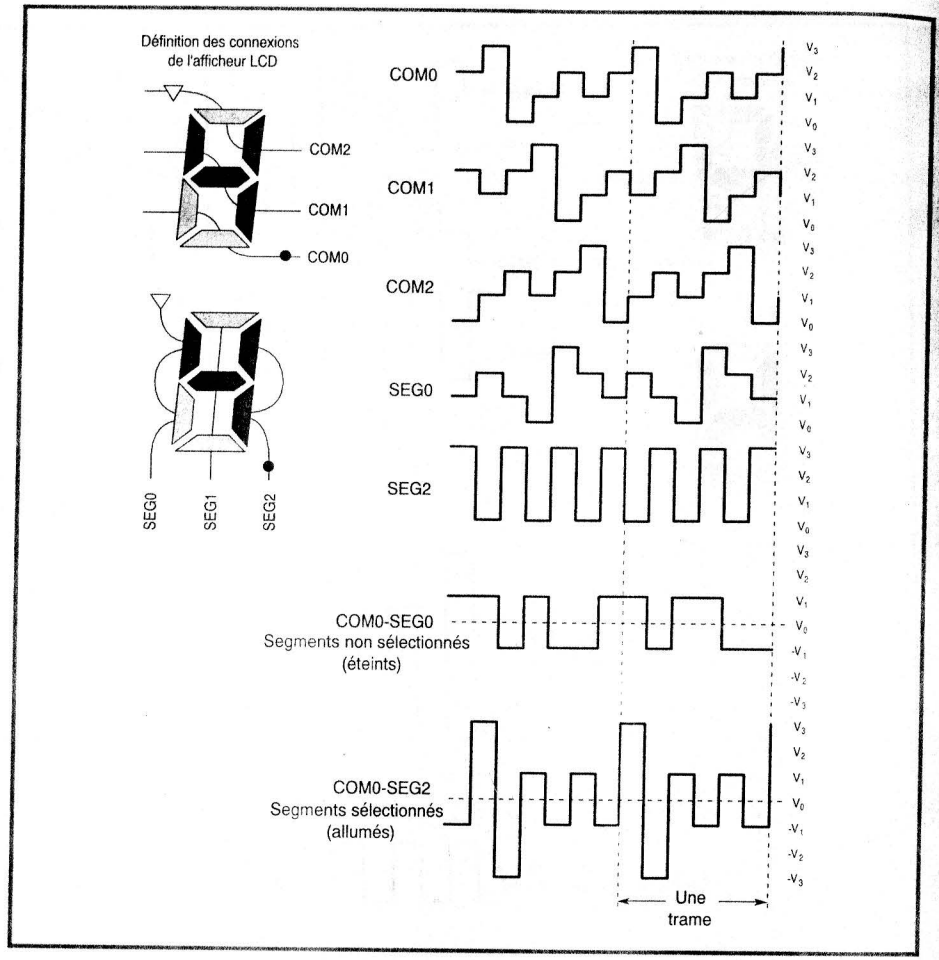


Figure 13.5 – Chronogrammes en mode multiplexage 1/3 et polarisation 1/3.

Cela étant précisé, la séquence d'initialisation à respecter pour utiliser l'interface est la suivante, qui découle naturellement des explications que nous vous avons fournies jusqu'à maintenant :

- Choisissez le taux de division de l'horloge trame au moyen des bits LP0 à LP3 du registre LCDPS.
- Configurez les pattes réellement utilisées par l'afficheur au moyen des bits appropriés du registre LCDSE.

© Dunod. La photocopie non autorisée est un délit.

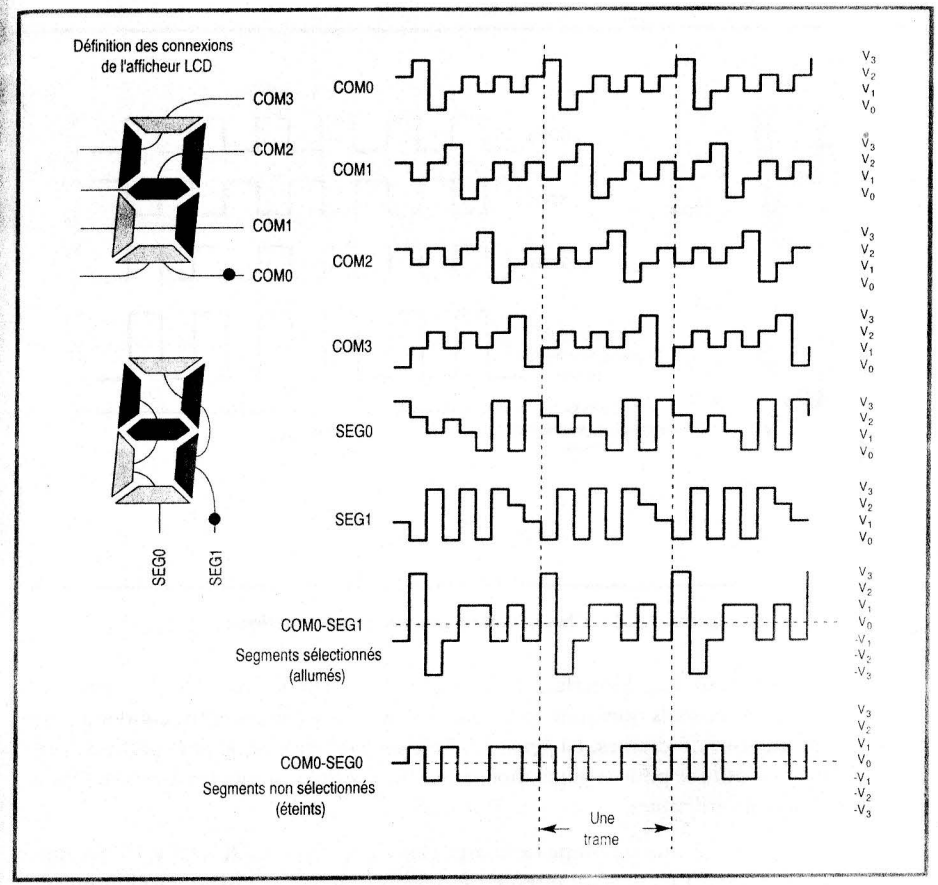
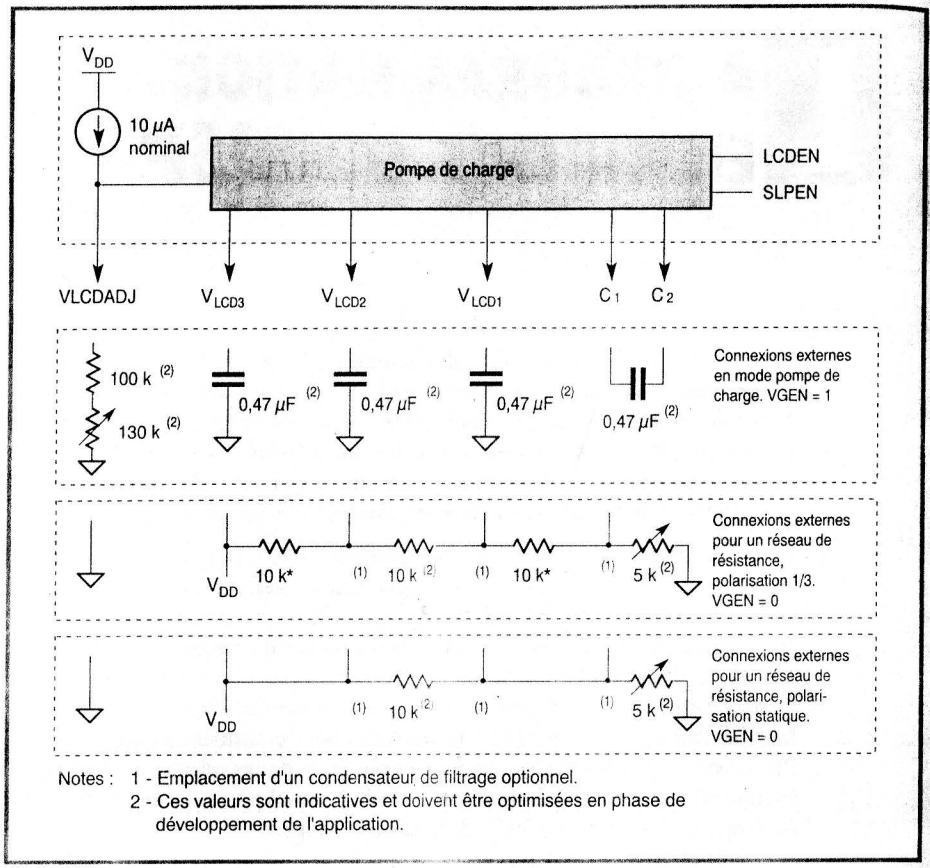


Figure 13.6 – Chronogrammes en mode multiplexage 1/4 et polarisation 1/3.

- Configurez l'interface en fonction du type d'afficheur utilisé en positionnant les bits suivants du registre LCDCON :
  - choix du mode de multiplexage avec LMUX0 et LMUX1 ;
  - source d'horloge avec CS0 et CS1 ;
  - générateur de tension avec ou sans pompe de charge grâce à VGEN ;
  - fonctionnement ou non en mode SLEEP grâce à SLPEN.
- Écrivez les valeurs initiales devant être affichées dans les registres LCDD00 à LCDD15.



Notes : 1 - Emplacement d'un condensateur de filtrage optionnel.
2 - Ces valeurs sont indicatives et doivent être optimisées en phase de développement de l'application.

Figure 13.7 – Schéma équivalent et mode d'utilisation du générateur de tensions de l'interface pour afficheurs LCD.

– Mettez à 0 le drapeau d'interruption LCDIF dans le registre PIRx et, si nécessaire, autorisez les interruptions en provenance de cette interface en positionnant le bit LCDIE dans le registre PIEx.

– Validez l'interface LCD et donc l'affichage, en positionnant le bit LCDEN.

Comme nous l'avons vu ci-dessus, l'interface peut ou non être autorisée à fonctionner en mode SLEEP. Si elle ne l'est pas, l'interface passe dans un mode à très faible consommation de courant en mode sommeil et arrête immédiatement toute activité au niveau de l'afficheur. De ce fait, et pour être certain que la trame d'affichage en cours se termine correctement, il faut veiller à exécuter l'instruction

SLEEP immédiatement après la fin d'une trame. Pour cela, l'utilisation de l'interruption en provenance de l'interface est quasiment indispensable puisque nous avons vu ci-dessus que c'était son rôle principal.

Si l'interface est autorisée à fonctionner en mode SLEEP, la seule précaution à prendre consiste à sélectionner comme source d'horloge, soit l'horloge RC interne, soit l'horloge provenant du *timer 1* puisque l'une comme l'autre continuent de fonctionner dans ce mode.

CHAPITRE 14

LA PROGRAMMATION EN CIRCUIT OU ICSP

Tous les microcontrôleurs PIC des familles 12Cxxx et 16Cxxx supportent la programmation en circuit appelée aussi ICSP pour *in circuit serial programming*. Ce mode de programmation particulier permet de programmer la mémoire du microcontrôleur alors que celui-ci est déjà installé dans l'application finale. Il est ainsi possible de stocker à l'avance des produits vierges et de les personnaliser au moment de la livraison en fonction des commandes des clients.

Ce mode de programmation permet aussi de mettre très facilement à jour des produits existants en remplaçant le programme contenu dans la mémoire par une version plus récente. Enfin, indépendamment de toute notion de programmation en circuit, cette façon de faire simplifie la réalisation des programmateurs, puisque l'on passe d'une programmation parallèle classique nécessitant de nombreuses liaisons à une programmation série qui se contente de cinq fils. Il est alors très facile de réaliser un programmateur universel pour quasiment tous les modèles de PIC existants, puisque seuls changent les positions de ces cinq fils sur les pattes des circuits en fonction de leur brochage, ainsi que quelques constantes en mémoire du programmateur, en fonction de la taille de la mémoire du PIC à programmer.

14.1 Mise en mode programmation

Pour faire passer un PIC en mode programmation, il faut maintenir ses lignes de ports parallèles RB6 et RB7 au niveau bas pendant que l'on fait monter la tension sur l'entrée de *reset* $\overline{\text{MCLR}}$ de V_{IL} à V_{IHH} (voir fiche technique de chaque circuit pour la valeur exacte de ce paramètre, généralement voisin de 13 V) et que la tension d'alimentation positive V_{DD} du circuit adopte la valeur de la tension de programmation indiquée elle aussi dans la fiche technique du circuit.

RB6 devient alors l'horloge de programmation et se comporte donc comme une entrée, alors que RB7 devient l'entrée/sortie série des données. Elle fonctionne en entrée pendant toute la phase de programmation proprement dite et en sortie

© Dunod. La photocopie non autorisée est un délit.

lors de la phase de vérification. RB6 et RB7 disposent de trigger de Schmitt en entrée, alors que RB7 est un buffer CMOS lorsqu'elle fonctionne en sortie.

Pendant toute la durée de la programmation, le *timer* chien de garde est automatiquement invalidé afin d'éviter qu'il ne génère un *reset* qui serait alors pour le moins indésirable !

14.2 Contraintes de la programmation en circuit

Si la programmation ne doit être réalisée que sur un programmateur, aucun problème ne se pose puisque ce dernier doit seulement piloter RB6, RB7, $\overline{MCLR}$ ainsi que la tension d'alimentation du PIC à programmer. Il suffit juste de respecter les niveaux de tension et chronogrammes indiqués dans les fiches techniques spécifiques des circuits.

Par contre, si vous souhaitez réellement pouvoir faire de la programmation en circuit, il faut le prévoir lors de la conception de l'application de façon à respecter les quelques règles que voici.

La *figure 14.1* montre un schéma de principe de la partie « programmation en circuit » d'une application et va servir de base à nos explications.

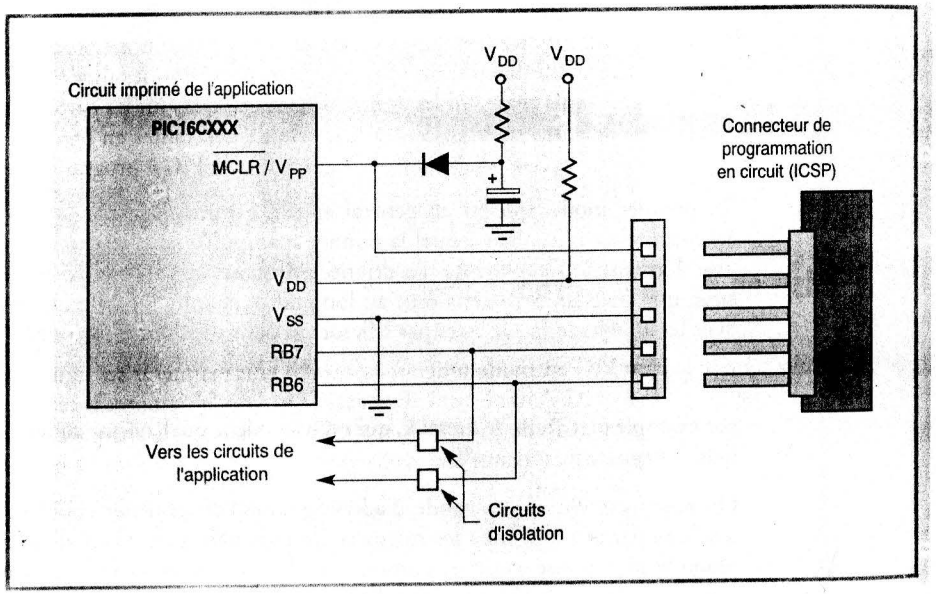


Figure 14.1 – Schéma de principe de la programmation en circuit.

La ligne $\overline{MCLR}$ est généralement connectée à une circuiterie de *reset* de type R-C ou similaire. Comme elle doit pouvoir être amenée à la tension V_{IHH} avec un temps de montée relativement court de surcroît, une circuiterie d'isolement doit être prévue, par exemple au moyen d'une diode comme schématisé sur la figure. Afin d'éviter que le seuil de la diode ne modifie de manière trop importante celui de la circuiterie de *reset*, cette dernière peut être avantageusement de type Schottky.

Les pattes RB6 et RB7, utilisées respectivement comme lignes d'horloge et de données pendant la programmation, doivent pouvoir être isolées de la circuiterie du reste de l'application. Le schéma de cette « isolation » dépend évidemment de l'application et aucun schéma général ne peut être fourni dans le cadre de cet ouvrage. Pour sa conception, notez bien que RB6 ne fonctionne qu'en entrée pendant la programmation, puisque c'est l'horloge de programmation, alors que RB7 doit pouvoir fonctionner en entrée et en sortie. En outre, comme ces deux lignes véhiculent des signaux logiques aux chronogrammes bien précis, il importe de minimiser les capacités parasites à leur niveau afin de ne pas dégrader les transitions des signaux logiques. Afin de simplifier la réalisation de cette partie de la schématique de la programmation en circuit, Microchip recommande d'essayer de respecter les indications suivantes, présentées par ordre de préférence :

- ne pas utiliser RB6 et RB7 dans l'application, ce qui est évidemment la solution idéale ;
- n'utiliser ces lignes qu'en sorties avec une charge capacitive minimale ;
- enfin, réaliser une circuiterie d'isolation conforme aux indications ci-dessus si les deux solutions précédentes n'ont pu être mises en œuvre.

Pour ce qui est de l'alimentation, il faut que le programmateur puisse piloter l'alimentation du microcontrôleur pendant toute la phase de programmation. En effet, d'une part la tension d'alimentation doit être égale à 5 V pendant que la programmation a lieu, d'autre part celle-ci doit pouvoir être réglée sur 2,7 et 4,5 V en phase de vérification de la programmation. Compte tenu des éléments associés au microcontrôleur sur l'application concernée, une circuiterie d'isolation peut donc être rendue nécessaire ou pas. Elle peut être réalisée au moyen d'un simple commutateur manuel si le processus de programmation en circuit n'a pas besoin d'être automatisé, ou faire appel à un commutateur électronique à diodes et/ou transistors dans le cas contraire.

Enfin, il importe de prendre des précautions au niveau de l'oscillateur d'horloge système du PIC car il faut absolument éviter que celle-ci puisse commencer à fonctionner avant que le circuit ne soit mis en mode programmation. En effet, le PC est utilisé comme compteur pendant la phase de programmation et il est censé partir de 000. Or, si l'horloge système a commencé à fonctionner, le PC n'est plus à 0 et la programmation de la mémoire de programme est réalisée avec un décalage qui la rend alors inutilisable.

Si l'oscillateur d'horloge est un modèle à quartz, ce problème ne se pose pas puisque, comme nous l'avons vu au chapitre 2, un délai de 1 024 cycles d'oscillateur est nécessaire avant que l'horloge système interne ne démarre. Par contre, si l'oscillateur est de type RC, ce délai se réduit à 4 cycles, ce qui est très court. Dans ce dernier cas, il est donc nécessaire de prévoir un moyen d'inhiber cet oscillateur pendant la phase de programmation en circuit, par exemple en déconnectant la résistance externe de cet oscillateur ou bien encore en reliant à la masse la patte OSC1 du microcontrôleur.

Compte tenu du fait que les lignes RB6 et RB7 véhiculent des signaux logiques mais aussi que $\overline{\text{MCLR}}$ doit changer de niveau avec des transitions rapides, le câble de liaison entre le connecteur de programmation en circuit placé sur l'application et le programmateur doit être maintenu aussi court que possible et doit être à faible capacité parasite.

CHAPITRE 15

ADRESSAGE ET JEU D'INSTRUCTIONS

Si vous avez déjà eu entre les mains un manuel de programmation Microchip, sa lecture vous a peut-être plongé dans un abîme de perplexité. En effet, alors que tous les fabricants de microcontrôleurs s'empressent de vous parler du jeu d'instructions et des modes d'adressage, ici on trouve bien le jeu d'instructions mais rien concernant les modes d'adressage.

Cette approche particulière s'explique par le fait que ces modes d'adressage sont très peu nombreux et font très souvent partie intégrante des instructions, comme nous le verrons dans un instant. Afin de ne pas trop dépayser ceux d'entre vous qui sont déjà habitués aux microcontrôleurs plus « classiques », nous allons cependant essayer ci-après d'en dresser une liste logique.

15.1 Les modes d'adressage

Le premier mode, qui est en général appelé « immédiat » par la majorité des fabricants, est celui dans lequel la donnée manipulée par l'instruction est codée avec l'instruction elle-même. La donnée en question s'appelle ici un « literal », nom que nous conserverons tout au long de ce chapitre afin de rester cohérent avec les appellations adoptées par Microchip dans tous ses documents.

```
MOVLW k
```

par exemple place ainsi le literal k, qui est une valeur quelconque codée sur 8 bits, dans le registre de travail W.

On rencontre ensuite un mode d'adressage que l'on pourrait appeler « direct » toujours par analogie avec les microcontrôleurs habituels. C'est d'ailleurs ici le mode le plus utilisé puisque, comme nous l'avons vu dans les chapitres précédents, la mémoire RAM est en fait divisée en registres spécifiques et en un ensemble de registres à usage général.

Ce mode consiste donc à coder le nom du ou des registres concernés directement dans l'instruction. Ainsi, par exemple :

```
MOVWF f
```

déplace le contenu du registre W dans le registre f. Le registre f est repéré par son « numéro » codé sur cinq ou sept bits, qui n'est rien d'autre en fait que l'adresse de l'octet de RAM correspondant, après exploitation si nécessaire du mécanisme de pagination mémoire.

On dispose également, comme sur tout microcontrôleur qui se respecte, d'un mode d'adressage de type bit qui permet de manipuler un bit individuel dans n'importe quel registre. Il est à noter que ce mode d'adressage ne s'utilise jamais seul mais est toujours couplé avec le mode d'adressage direct vu ci-dessus. Ainsi :

```
BCF f,b
```

met à zéro le bit numéro b du registre f. f est codé comme indiqué ci-dessus, avec son numéro de registre sur 5 ou 7 bits, et b n'est autre que le numéro du bit de ce registre, codé lui sur 3 bits seulement puisqu'il ne peut varier que de 0 à 7.

Enfin, le dernier mode d'adressage que certains auront tendance à appeler le mode le plus puissant par analogie avec des microcontrôleurs classiques, est le mode indirect dont nous avons déjà parlé lors de la présentation des registres f0 ou INDF et f4 ou FSR.

Rappelons donc rapidement que f4 est le FSR ou registre de sélection de registre dans lequel on place le numéro du registre adressé. La seule « bizarrerie » des PIC est le mode de notation correspondant qui utilise le registre f0 ou INDF sous la forme par exemple :

```
MOVWF f0
```

Cette instruction déplace le contenu du registre de travail W dans le registre pointé par le registre f4 ou FSR. La notation f0 ne sert ici qu'à signaler l'adressage indirect, et donc l'utilisation du registre FSR comme pointeur.

Même si cela vous semble maigre, surtout si vous êtes habitué à des microcontrôleurs tel que le 68HC11 par exemple, ce sont là tous les modes d'adressage des PIC 12Cxxx et 16Cxxx. Compte tenu de l'efficacité de la structure des circuits et de leur jeu d'instructions, cela permet tout de même de faire du beau travail.

Notez, avant de conclure cette partie, que nous n'avons pas oublié le traditionnel mode d'adressage dit « relatif » qui permet, dans les architectures classiques, les sauts lors des branchements conditionnels par exemple. La lecture du descriptif des instructions de branchement vous permettra de comprendre pourquoi ce mode n'existe pas vraiment ici.

15.2 Le jeu d'instructions

Afin de vous fournir un ouvrage aussi pratique que possible, chaque instruction vous est présentée dans les pages qui suivent avec :

- sa syntaxe, tenant compte des notations précisées ci-après ;
- son codage sur 12 bits dans le cas des 12Cxxx ;
- son codage sur 14 bits dans le cas des 16Cxxx ;
- le nombre de mots et le nombre de cycles machine utilisés. Vous pouvez ainsi constater que le principe fondamental de l'architecture RISC est bien respecté avec une instruction par cycle, sauf dans le cas de très rares exceptions ;
- la représentation synoptique de l'opération réalisée par l'instruction ;
- les bits du registre d'état qui sont affectés par cette opération ;
- enfin, nos commentaires, issus de la documentation du fabricant ou de notre expérience personnelle lorsque ses explications étaient par trop nébuleuses !

Remarquez que le codage des instructions ne vous intéresse pas pour ce qui est de la programmation proprement dite, puisqu'il est géré par l'assembleur, mais il vous permet de comprendre comment l'on arrive à coder des instructions et les modes d'adressage qui suivent sur seulement 12 ou 14 bits ! En outre, cela permet également de comprendre les limites de tailles imposées à certaines données, limites directement dues au nombre de bits utilisables.

La notation adoptée pour les données et adresses manipulées par les instructions est fort simple et est la suivante :

- f représente un numéro de registre, codé sur cinq bits dans le cas des 12Cxxx et sept bits dans le cas des 16Cxxx.
- b représente un numéro de bit, codé sur trois bits. Le bit zéro est toujours le bit de poids le plus faible comme il se doit en notation logique cohérente (ce n'est hélas pas le cas pour tout le monde !).
- k représente une donnée, le fameux « literal » évoqué ci-dessus. Son codage en nombre de bits est variable selon l'instruction.

Nous vous invitons à lire en détail le descriptif de ces instructions pour bien apprécier les possibilités des circuits, mais nous tenons dès à présent à vous faire remarquer quelques particularités intéressantes.

Un certain nombre d'instructions (ADDWF, ANDWF, etc.) utilisent une notation spéciale présentée sous la forme :

```
ADDWF f,d
```

f est le numéro de registre, ce qui est déjà connu ; par contre d peut prendre la valeur 0 ou 1 et change le comportement de l'instruction. Si d est à 0, le résultat est placé dans le registre W alors que si d est à 1, le résultat est placé dans le registre f. Les instructions de ce type sont donc en fait doubles puisque, selon la valeur de d, elles font :

- W opération f → W si d = 0 ou ;
- W opération f → f si d = 1.

Le second point que nous aimerions vous faire remarquer concerne les instructions de branchement conditionnel. Prenons par exemple :

```
BTFSC f,b
```

Vu sa notation, vous savez déjà qu'elle teste le bit b du registre f. Comme c'est un *bit test skip if clear*, elle réalise un saut (*skip*) si le bit est à zéro (*clear*). Mais cela se passe de la façon suivante :

- si le bit est à 1, donc si la condition testée *n'est pas réalisée*, le programme continue son déroulement normal en séquence ;
- si le bit est à 0, donc si la condition testée *est réalisée*, le programme saute l'instruction qui suit le BTFSC dans le programme.

Si donc vous voulez, comme c'est généralement le cas, vous brancher à une autre partie du programme selon le résultat du test, vous adopterez généralement la syntaxe suivante :

```
Instruction de test et de branchement
GOTO k ou CALL k
```

En procédant de la sorte, si la condition est fausse, vous allez exécuter la branche de programme appelée par le GOTO ou le CALL, alors que si la condition est vraie, vous continuez le programme avec l'instruction qui suit le GOTO ou le CALL, puisque vous avez justement sauté cette ligne dans ce cas.

Notez que cette approche, qui peut sembler curieuse à première vue, est en réalité très pratique pour agir sur des bits de registres en fonction de conditions externes. Ainsi, si un interrupteur est raccordé sur une ligne de port parallèle et que l'on doive, en fonction de son état, positionner un relais commandé par une autre ligne de port parallèle, il suffira de faire :

```
BTFSC f,b
```

où b est le numéro de bit correspondant à l'interrupteur sur le port concerné.

```
BCF f,b
```

où b est le numéro du bit activant le relais sur le port concerné.

Ce traitement particulier des instructions de test permet de comprendre pourquoi nous vous indiquions, en début de chapitre, que l'adressage relatif n'existait pas vraiment.

Les autres instructions sont plus classiques, mais arrêtons-nous tout de même un instant sur CALL qui est un appel de sous-programme et GOTO qui est un branchement inconditionnel.

Dans le cas des CALL et GOTO, notez que l'adresse appelée semble codée sur un nombre de bits insuffisant. En fait, elle est bien codée sur 12 ou 14 bits mais les bits « manquants » proviennent soit de PA0 dans le cas des PIC 12Cxxx, soit de PCLATH dans le cas des 16Cxxx. Revoyez si nécessaire le chapitre 3 qui présente le mécanisme de pagination mémoire. Il faut donc songer à positionner PA0 ou à charger PCLATH correctement au préalable si vous ne voulez pas avoir de surprise désagréable !

Lorsque vous aurez lu et assimilé ces différentes instructions, vous serez en droit d'apprécier les tableaux synthétiques qui résument l'essentiel de ce qu'il faut savoir à leur sujet, tableaux que nous vous proposons en fin de chapitre.

15.3 Compatibilité ascendante

En lisant la description qui va suivre, vous constaterez que nous déconseillons d'utiliser deux instructions qui sont OPTION et TRIS avec les circuits de la famille 16Cxxx.

La raison d'être de cette apparente anomalie est fort simple. Sur ces circuits ces deux instructions sont devenues des instructions « fantômes » et n'ont été maintenues que pour assurer la compatibilité du code source avec les programmes écrits pour les « vieux » 16C5X.

En effet, compte tenu de la structure interne des circuits 16Cxxx, les registres OPTION et TRIS sont directement accessibles comme tous les autres registres. Il n'y a donc plus lieu de faire appel à des instructions spécifiques pour les manipuler comme c'était le cas avec les PIC 16C5x. Les instructions OPTION et TRIS n'ont donc plus aucune raison d'être sur ces circuits, puisqu'elles peuvent être remplacées par un MOVWF avec la bonne valeur de f.

Par contre, ces deux instructions doivent être utilisées avec les PIC 12Cxxx qui, sur ce plan, fonctionnent comme les « vieux » 16C5x.

15.4 Description détaillée des instructions

Après ce préambule indispensable, voici la description détaillée du jeu d'instructions complet des PIC 12Cxxx et 16Cxxx, classées comme il se doit par ordre alphabétique.

ADDLW (*add literal to W*)

Syntaxe : ADDLW k

Codage : 11 111x kkkk kkkk

Mots, cycles : 1, 1

Opération : W + k → W

Bit d'état : C, DC, Z

Description : ajoute le contenu de W au *literal* k et place le résultat dans W.

Attention ! Cette instruction n'existe pas sur les PIC 12Cxxx.

ADDWF (*ADD W to F*)

Syntaxe : ADDWF f,d

Codage : 0001 11 df ffff

00 0111 dfff ffff

Mots, cycles : 1, 1

Opération : W + f → f si d = 1 ou W + f → W si d = 0

Bit d'état : C, DC, Z

Description : ajoute le contenu de W et le contenu de f et place le résultat dans W si d = 0 et dans f si d = 1.

ANDLW (*AND literal and W*)

Syntaxe : ANDLW k

Codage : 1110 kkkk kkkk

11 1001 kkkk kkkk

Mots, cycles : 1, 1

Opération : W ET k → W

Bits d'état : Z

Description : effectue un ET logique entre le contenu de W et le *literal* k et place le résultat dans W.

ANDWF (*AND W with F*)

Syntaxe : ANDWF f,d

Codage : 0001 01df ffff

00 0101 dfff ffff

Mots, cycles : 1, 1

Opération : W ET f → f si d = 1 ou W + f → W si d = 0

Bit d'état : Z

Description : effectue un ET logique entre le contenu de W et le contenu de f et place le résultat dans W si d = 0 et dans f si d = 1.

BCF (*bit clear F*)

Syntaxe : BCF f,b

Codage : 0100 bbbf ffff

01 00bb bfff ffff

Mots, cycles : 1, 1

Opération : 0 → b(f)

Bit d'état : aucun

Description : met à 0 le bit numéro b de f.

BSF (*bit set F*)

Syntaxe : BSF f,b

Codage : 0101 bbbf ffff

01 01bb bfff ffff

Mots, cycles : 1, 1

Opération : 1 → b(f)

Bit d'état : aucun

Description : met à 1 le bit numéro b de f.

BTFSC (*bit test, skip if clear*)

Syntaxe : BTFSC f,b

Codage : 0110 bbbf ffff

01 10bb bfff ffff

Mots, cycles : 1, 1 ou 2 (voir description)

Opération : saut si b(f) = 0

Bit d'état : aucun

Description : si le bit numéro b de f est nul, l'instruction qui suit celle-ci est ignorée et traitée comme un NOP. Dans ce cas, et dans ce cas seulement, le BTFSC demande deux cycles pour s'exécuter.

BTFSS (*bit test, skip if set*)

Syntaxe : BTFSS f,b

Codage : 0111 bbbf ffff

01 11bb bfff ffff

Mots, cycles : 1, 1 ou 2 (voir description)

Opération : saut si b(f) = 1

Bit d'état : aucun

Description : Si le bit numéro b de f est à 1, l'instruction qui suit celle-ci est ignorée et traitée comme un NOP. Dans ce cas, et dans ce cas seulement, le BTFSS demande deux cycles pour s'exécuter.

CALL (*subroutine call*)

Syntaxe : CALL k

Codage : 1001 kkkk kkkk

10 0kkk kkkk kkkk

Mots, cycles : 1, 2

Opération :

– dans le cas des 12Cxxx : PC+1 → pile, k → PC (0-7),0 → PC (8), PA0 → PC (9)

– dans le cas des 16Cxxx : PC+1 → pile, k → PC (0-10), PCLATH (3,4) → PC (11,12)

Bit d'état : aucun

Description : sauvegarde l'adresse de retour sur la pile puis appelle le sous-programme situé à l'adresse chargée dans le PC.

Attention ! le mode de calcul de l'adresse diffère selon la famille de PIC utilisée. Il faut également positionner correctement soit PA0 (PIC 12Cxxx), soit le registre PCLATH (PIC 16Cxxx) avant d'effectuer le CALL.

CLRF (*clear F*)

Syntaxe : CLRF f

Codage : 0000 011f ffff

00 0001 1fff ffff

Mots, cycles : 1, 1

Opération : 00 → f

Bit d'état : Z

Description : met le contenu de f à 0 et positionne le bit Z.

CLRW (*clear W register*)

Syntaxe : CLRW

Codage : 0000 0100 0000

00 0001 0xxx xxxx

Mots, cycles : 1, 1

Opération : 00 → W

Bit d'état : Z

Description : met le registre W à 0 et positionne le bit Z.

CLRWDT (*clear watchdog timer*)

Syntaxe : CLRWDT

Codage : 0000 0000 0100

00 0000 0110 0100

Mots, cycles : 1, 1

Opération : 00 → WDT et 0 → prédiviseur du *timer*

Bit d'état : 1→ TO et 1 → PD

Description : met à 0 le registre de comptage du *timer* chien de garde ainsi que le prédiviseur.

COMF (*complement F*)

Syntaxe : COMF f,d

Codage : 0010 01df ffff

0 1001 dfff ffff

Mots, cycles : 1, 1

Opération : /f → f si d = 1 ou/f → W si *d* = 0

Bit d'état : Z

Description : complémente f bit à bit. Le résultat est placé à nouveau dans f si *d* = 1 ou dans W si *d* = 0 (dans ce cas f reste inchangé).

DECF (*decrement F*)

Syntaxe : DECF f,d

Codage : 0000 11df ffff

00 0011 dfff ffff

Mots, cycles : 1, 1

Opération : f – 1 → f si d = 1 ou f – 1 → W si d = 0

Bit d'état : Z

Description : diminue le contenu de f d'une unité. Le résultat est placé à nouveau dans f si d = 1 ou dans W si d = 0 (dans ce cas f reste inchangé).

DECFSZ (*decrement F, skip if zero*)

Syntaxe : DECFSZ f,d

Codage : 0010 11df ffff

00 1011 dfff ffff

Mots, cycles : 1, 1(2)

Opération : f – 1 → f si d = 1 ou f – 1 → W si d = 0 et saut si f – 1 = 0

Bit d'état : aucun

Description : diminue le contenu de f d'une unité. Le résultat est placé à nouveau dans f si d = 1 ou dans W si d = 0 (dans ce cas f reste inchangé). Si le résultat est nul l'instruction suivante est ignorée et, dans ce cas, cette instruction dure deux cycles.

GOTO (branchement inconditionnel)

Syntaxe : GOTO k

Codage : 101k kkkk kkkk

10 1kkk kkkk kkkk

Mots, cycles : 1, 2

Opération :

– dans le cas des 12Cxxx : k -> PC (0-8), PA0 -> PC (9)

– dans le cas des 16Cxxx : k -> PC (0-10), PCLATH (3,4) ->PC (11,12)

Bit d'état : aucun

Description : appelle le sous-programme situé à l'adresse chargée dans le PC.

Attention ! le mode de calcul de l'adresse diffère selon la famille de PIC utilisée. Il faut également positionner correctement soit PA0 (PIC 12Cxxx), soit le registre PCLATH (PIC 16Cxxx) avant d'effectuer le CALL.

INCF (*increment F*)

Syntaxe : INCF f,d

Codage : 0010 10df ffff

00 1010 dfff ffff

Mots, cycles : 1, 1

Opération : f + 1 → f si d = 1 ou f + 1 → W si d = 0

Bit d'état : Z

Description : augmente le contenu de f d'une unité. Le résultat est placé à nouveau dans f si d = 1 ou dans W si d = 0 (dans ce cas f reste inchangé).

INCFSZ (*increment F, skip if zero*)

Syntaxe : INCFSZ f,d

Codage : 0011 11df ffff

00 1111 dfff ffff

Mots, cycles : 1, 1(2)

Opération : f + 1 → f si d = 1 ou f + 1 → W si d = 0 et saut si f + 1 = 0

Bit d'état : aucun

Description : augmente le contenu de f d'une unité. Le résultat est placé à nouveau dans f si d = 1 ou dans W si d = 0 (dans ce cas f reste inchangé). Si le résultat est nul l'instruction suivante est ignorée, dans ce cas, cette instruction dure deux cycles.

IORLW (*inclusive OR literal with W*)

Syntaxe : IORLW k

Codage : 1101 kkkk kkkk

11 1000 kkkk kkkk

Mots, cycles : 1, 1

Opération : W OU k → W

Bit d'état : Z

Description : effectue un OU logique inclusif entre le contenu de W et le literal k et place le résultat dans W.

IORWF (*inclusive OR W with F*)

Syntaxe : IORWF f,d

Codage : 0001 00df ffff

00 0100 dfff ffff

Mots, cycles : 1, 1

Opération : W OU f → f si d = 1 ou W OU f → W si d = 0

Bit d'état : Z

Description : effectue un OU logique inclusif entre le contenu de W et le contenu de f et place le résultat dans f si d = 1 ou dans W si d = 0.

MOVF (*move F*)

Syntaxe : MOVF f,d

Codage : 0010 00df ffff

00 1000 dfff ffff

Mots, cycles : 1, 1

Opération : f → f si d = 1 ou f → W si d = 0

Bit d'état : Z

Description : déplace le contenu de f dans f si d = 1 ou dans W si d = 0.

Attention ! le déplacement de f dans f qui semble *a priori* inutile permet en fait de tester le contenu de f par rapport à zéro, puisque cette instruction positionne le bit Z.

MOVLW (*move literal to W*)

Syntaxe : MOVLW k

Codage : 1100 kkkk kkkk

11 00xx kkkk kkkk

Mots, cycles : 1, 1

Opération : k → W

Bit d'état : aucun

Description : charge W avec le *literal* k.

MOVWF (*move W to F*)

Syntaxe : MOVWF f

Codage : 0000 001f ffff

00 0000 1fff ffff

Mots, cycles : 1, 1

Opération : W → f

Bit d'état : aucun

Description : charge f avec le contenu de W.

NOP (*no operation*)

Syntaxe : NOP

Codage : 0000 0000 0000

00 0000 0xx0 0000

Mots, cycles : 1, 1

Opération : néant

Bit d'état : aucun

Description : ne fait que consommer du temps machine (un cycle dans ce cas) comme tout NOP qui se respecte.

OPTION (*load option register*)

Syntaxe : OPTION

Codage : 0000 0000 0010

00 0000 0110 0010

Mots, cycles : 1, 1

Opération : W → OPTION

Bit d'état : aucun

Description : charge le registre OPTION avec le contenu de W.

Attention ! cette instruction ne doit pas être utilisée sur d'autres circuits que les PIC 12Cxxx et sur les « vieux » 16C5x. Elle est néanmoins correctement interprétée par les circuits de la famille 16Cxxx afin d'assurer une compatibilité ascendante.

RETFIE (*return from interrupt*)

Syntaxe : RETFIE

Codage : 00 0000 0000 1001

Mots, cycles : 1, 2

Opération : Pile → PC, 1 → GIE

Bit d'état : aucun

Description : charge le PC avec la valeur qui se trouve au sommet de la pile, assurant ainsi un retour d'interruption. Positionne à 1 le bit GIE afin d'autoriser à nouveau la prise en compte des interruptions.

Attention ! cette instruction dure deux cycles et n'existe pas sur les PIC 12Cxxx.

RETLW (*return literal to W*)

Syntaxe : RETLW k

Codage : 1000 kkkk kkkk

11 01xx kkkk kkkk

Mots, cycles : 1, 2

Opération : k → W, pile → PC

Bit d'état : Aucun

Description : charge W avec le *literal* k, puis charge le PC avec la valeur qui se trouve au sommet de la pile, effectuant ainsi un retour de sous programme.

Attention ! cette instruction dure deux cycles.

RETURN (*return from subroutine*)

Syntaxe : RETURN

Codage : 00 0000 0000 1000

Mots, cycles : 1, 2

Opération : Pile → PC

Bit d'état : aucun

Description : charge le PC avec la valeur qui se trouve au sommet de la pile, effectuant ainsi un retour de sous-programme. Il s'agit d'un RETLW simplifié.

Attention ! cette instruction dure deux cycles. Elle n'existe pas sur les PIC 12Cxxx.

RLF (*rotate left F through carry*)

Syntaxe : RLF f,d

Codage : 0011 01df ffff

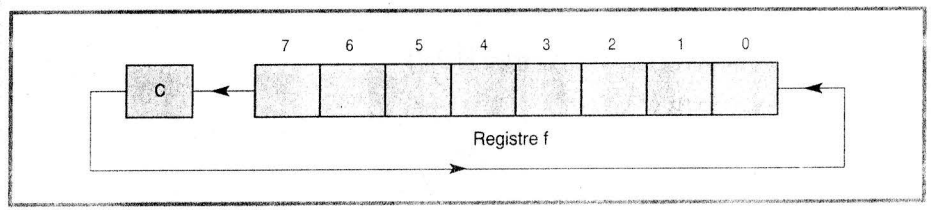
00 1101 dfff ffff

Mots, cycles : 1, 1

Opération : voir description

Bit d'état : C

Description : rotation à gauche d'un bit du contenu de f en passant par le bit de retenue C. Si d = 1 le résultat est placé dans f, si d = 0 le résultat est placé dans W.



RRF (*rotate right F through carry*)

Syntaxe : RRF f,d

Codage : 0011 00df ffff

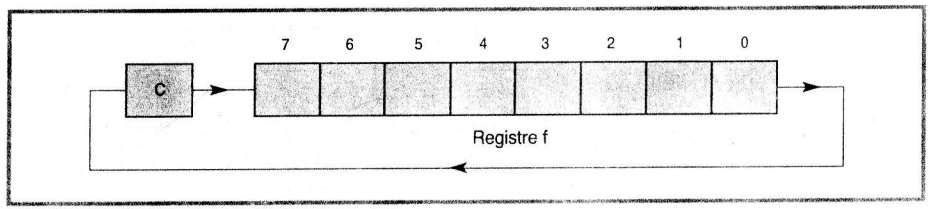
00 1100 dfff ffff

Mots, cycles : 1, 1

Opération : voir description

Bit d'état : C

Description : rotation à droite d'un bit du contenu de f en passant par le bit de retenue C. Si d = 1 le résultat est placé dans f, si d = 0 le résultat est placé dans W.



SLEEP (*sleep*)

Syntaxe : SLEEP

Codage : 0000 0000 0011

00 0000 0110 0011

Mots,cycles : 1, 1

Opération : 0 → PD, 1 → TO, 00 → WDT, 0 → prédiviseur du WDT

Bits d'état : TO, PD

Description : place le circuit en mode sommeil avec arrêt de l'oscillateur.

Attention ! consultez le chapitre 5 dans lequel le mode sommeil des PIC est décrit en détail.

SUBLW (*substract W from literal*)

Syntaxe : SUBLW k

Codage : 11 110x kkkk kkkk

Mots, cycles : 1, 1

Opération : k – W → W

Bit d'état : C, DC, Z

Description : soustrait le contenu de W du *literal* k et place le résultat dans W. Soustraction par la méthode du complément à deux.

Attention ! cette instruction n'existe pas sur les PIC 12Cxxx.

SUBWF (*substract W from F*)

Syntaxe : SUBWF f,d

Codage : 0000 10df ffff

00 0010 dfff ffff

Mots, cycles : 1, 1

Opération : f – W → f si d = 1 ou f – W → W si d = 0

Bit d'état : C, DC, Z

Description : soustrait le contenu de W du contenu de f et place le résultat dans W si d = 0 et dans f si d = 1. Soustraction par la méthode du complément à deux.

SWAPF (*swap F*)

Syntaxe : SWAPF f,d

Codage : 0011 10df ffff

00 1110 dfff ffff

Mots, cycles : 1, 1

Opération : f(0-3) → f(4-7) et f(4-7) → f(0-3) résultat → f ou W selon d

Bit d'état : aucun

Description : échange les quatre bits de poids forts avec les quatre bits de poids faibles de f et place le résultat dans f si d = 1 ou dans W si d = 0.

TRIS (*load TRIS register*)

Syntaxe : TRIS f

Codage : 0000 0000 0fff

00 0000 0110 0fff

Mots, cycles : 1, 1

Opération : W → registre TRIS du port f

Bits d'état : aucun

Description : charge le contenu de W dans le registre TRIS du port f.

Attention ! cette instruction ne doit pas être utilisée sur d'autres circuits que les PIC 12Cxxx et les « vieux » 16C5x. Elle est néanmoins correctement interprétée par les circuits de la famille 16Cxxx afin d'assurer une compatibilité ascendante.

XORLW (*exclusive OR literal with W*)

Syntaxe : XORLW k

Codage : 1111 kkkk kkkk

11 1010 kkkk kkkk

Mots, cycles : 1, 1

Opération : W OU EXCLUSIF k → W

Bit d'état : Z

Description : effectue un OU logique exclusif entre le contenu de W et le *literal* k et place le résultat dans W.

XORWF (*exclusive OR W with F*)

Syntaxe : XORWF f,d

Codage : 0001 10df ffff

00 0110 dfff ffff

Mots, cycles : 1, 1

Opération : W OU EXCL. f → f si d = 1 ou W OU EXCL. f → W si d = 0

Bit d'état : Z

Description : effectue un OU logique exclusif entre le contenu de W et le contenu de f et place le résultat dans f si d = 1 ou dans W si d = 0.

15.5 Tableaux synthétiques

Une fois que vous aurez commencé à programmer les microcontrôleurs PIC, la description détaillée qui précède vous deviendra bien vite inutile et seul un résumé des principales données de chaque instruction vous sera nécessaire. C'est dans ce but que nous vous proposons de découvrir maintenant deux *tableaux de synthèse 15.1* et *15.2* des instructions des microcontrôleurs PIC : un pour les circuits à mots de 12 bits que sont les « petits » 12Cxxx et les « vieux » 16C5x et un pour les circuits à mots de 14 bits que sont les 16Cxxx.

Tableau 15.1 – Tableau synthétique des instructions des PIC à mots de 12 bits (certains 12Cxxx et les 16C5x).

Instructions sur les registres (en octets)

(11-6)	(5)	(4-0)
OPCODE	d	f(FILE #)

d = 0 à destination de W
d = 1 à destination de f

Instructions-Binaire (Hex)				Nom	Mnémonique	Opérandes	Opération	Bit d'état	Notes
0001	11df	ffff	1Cf	Add W and f	ADDWF	f, d	W + f → d	C,DC,Z	1,2,4
0001	01df	ffff	14f	AND W and f	ANDWF	f, d	W & f → d	Z	2,4
0000	011f	ffff	06f	Clear f	CLRF	f	0 → f	Z	4
0000	0100	0000	040	Clear W	CLRW	-	0 → W	Z	
0010	01df	ffff	24f	Complement f	COMF	f, d	$\overline{f}$ → d	Z	2,4
0000	11df	ffff	0Cf	Decrement f	DECF	f, d	f-1 → d	Z	2,4
0010	11df	ffff	2Cf	Decrement f,Skip if Zero	DECFSZ	f, d	f-1 → d,Skip if Zero	None	2,4
0010	10df	ffff	28f	Increment f	INCF	f, d	f+1 → d	Z	2,4
0011	11df	ffff	3Cf	Increment f,Skip if Zero	INCFSZ	f, d	f+1 → d,Skip if Zero	None	2,4
0001	00df	ffff	10f	Inclusive OR Wand f	IORWF	f, d	W v f → d	Z	2,4
0010	00df	ffff	20f	Move f	MOVF	f, d	f → d	Z	2,4
0000	001f	ffff	02f	Move W to f	MOVWF	f	W → f	None	1,4
0000	0000	0000	000	No Operation	NOP	-	-	None	
0011	01df	ffff	34f	Rotate left f	RLF	f, d	f(n) → d(n+1), C → d(0), f(7) → C	C	2,4
0011	00df	ffff	30f	Rotate right f	RRF	f, d	f(n) → d(n-1), C → d(7), f(0) → C	C	2,4
0000	10df	ffff	08f	Substract W from f	SUBWF	f, d	f-W → d [f + $\overline{W}$ + 1 → d]	C,DC,Z	1,2,4
0011	10df	ffff	38f	Swap halves f	SWAPF	f, d	f(0-3) ↔ f (4-7) → d	None	2,4
0001	10df	ffff	18f	Exclusive OR Wand f	XORWF	f, d	W ⊕ f → d	Z	2,4

Instructions sur les registres (bit par bit)

(11-8)	(7-5)	(4-0)
OPCODE	d(bit #)	f(FILE #)

Instructions-Binaire (Hex)				Nom	Mnémonique	Opérandes	Opération	Bit d'état	Notes
0100	bbbf	ffff	4bf	Bit Cear f	BCF	f, b	0 → f(b)	None	2,4
0101	bbbf	ffff	5bf	Bit Set f	BSF	f, b	1 → f(b)	None	2,4
0110	bbbf	ffff	6bf	Bit Test f, Skip if Clear	BTFSC	f, b	Test bit (b) in file (f): Skip if Clear	None	
0111	bbbf	ffff	7bf	Bit Test f, Skip if Set	BTFSS	f, b	Test bit (b) in file (f): Skip if Set	None	

Instructions de contrôle

(11-8)	(7-0)
OPCODE	k(LITERAL)

Instructions-Binaire (Hex)				Nom	Mnémonique	Opérandes	Opération	Bit d'état	Notes
1110	kkkk	kkkk	Ekk	AND Literal and W	ANDLW	k	k & W → W	Z	
1001	kkkk	kkkk	9kk	Call subroutine	CALL	k	PC + 1 → Stack, k → PC	None	1
0000	0000	0100	004	Clear Watchdog timer	CLRWDT	-	0 → WDT (and prescaler, if assigned)	TO, PD	
101k	kkkk	kkkk	Akk	Go to address (k is 9 bit)	GOTO	k	k → PC (9 bits)	None	
1101	kkkk	kkkk	Dkk	Incl. OR Literal and W	IORLW	k	kv W → W	Z	
1100	kkkk	kkkk	Ckk	MOVE Literal to W	MOVLW	k	k → W	None	
0000	0000	0010	002	Load OPTION register	OPTION	-	W → OPTION register	None	
1000	kkkk	kkkk	8kk	Return,place Literal in W	RETLW	k	k → W, Stack → PC	None	
0000	0000	0011	003	Go into standby mode	SLEEP	-	0 → WDT, stop oscillator	TO, PD	
0000	0000	0fff	00f	Tristate port f	TRIS	f	W → I/O control register f	None	3
1111	kkkk	kkkk	Fkk	Excl. OR Literal and W	XORLW	k	k ⊕ W → W	Z	

Tableau 15.2 – Tableau synthétique des instructions des PIC à mots de 14 bits (certains 12Cxxx et tous les 16Cxxx).

Instructions sur les registres (en octets)

13	8	7	6	0
OPCODE		d	f(FILE #)	

d = 0 = destination de W d = 1 = destination de f
f = registre d'adresse, 7 bits

Instructions-Binaire	(Hex)	Nom	Mnémonique	Opérandes	Opération	Bit d'état	Notes
00 0111 dfff ffff	07ff	Add W and f	ADDWF	f, d	W + f → d	C,DC,Z	2,3
00 0101 dfff ffff	05ff	AND W and f	ANDWF	f, d	W & f → d	Z	2,3
00 0001 1fff ffff	018f	Clear f	CLRF	f	0 → f	Z	3
00 0001 0XXX XXXX	0100	Clear W	CLRW	-	0 → W	Z	
00 1001 dfff ffff	09ff	Complement f	COMF	f, d	$\overline{f}$ → d	Z	2,3
00 0011 dfff ffff	03ff	Decrement f	DECF	f, d	f-1 → d	Z	2,3
00 1011 dfff ffff	0Bff	Decrement f,Skip if Zero	DECFSZ	f, d	f-1 → d,Skip if Zero	None	2,3
00 1010 dfff ffff	0Aff	Increment f	INCF	f, d	f+1 → d	Z	2,3
00 1111 dfff ffff	0Fff	Increment f,Skip if Zero	INCFSZ	f, d	f+1 → d,Skip if Zero	None	2,3
00 0100 dfff ffff	04ff	Inclusive OR W and f	IORWF	f, d	W v f → d	Z	2,3
00 1000 dfff ffff	08ff	Move f	MOVF	f, d	f → d	Z	2,3
00 0000 1fff ffff	008f	Move W to f	MOVWF	f	W → f	None	3
00 0000 0XX0 0000	0000	No Operation	NOP	-	-	None	
00 1101 dfff ffff	0Dff	Rotate left f	RLF	f, d	f(n) → d(n+1), C → d(0), f(7) → C	C	2,3
00 1100 dfff ffff	0Cff	Rotate right f	RRF	f, d	f(n) → d(n-1), C → d(7), f(0) → C	C	2,3
00 0010 dfff ffff	02ff	Substract W from f	SUBWF	f, d	f-W → d [f + $\overline{W}$ + 1 → d]	C,DC,Z	2,3
00 1110 dfff ffff	0Eff	Swap halves f	SWAPF	f, d	f(0-3) ↔ f (4-7) → d	None	2,3
00 0110 dfff ffff	06ff	Exclusive OR W and f	XORWF	f, d	W ⊕ f → d	Z	2,3

Instructions sur les registres (bit par bit)

13	10	9	7	6	0
OPCODE		b(bit #)		f(FILE #)	

b = bit d'adresse, 3 bits f = registre d'adresse, 7 bits

01 00bb bfff ffff	1bff	Bit Cear f	BCF	f, b	0 → f(b)	None	2,3
01 01bb bfff ffff	1bff	Bit Set f	BSF	f, b	1 → f(b)	None	2,3
01 10bb bfff ffff	1bff	Bit Test f, Skip if Clear	BTFSC	f, b	Test bit (b) in file (f): Skip if Clear	None	
01 11bb bfff ffff	1bff	Bit Test f, Skip if Set	BTFSS	f, b	Test bit (b) in file (f): Skip if Set	None	

Instructions de contrôle

13	8	7	0
OPCODE		k(LITERAL)	

k = 8 bits, valeur immédiate

11 111X kkkk kkkk	3Ekk	Add literal to W	ADDLW	k	k + W → W	C,DC,Z	
11 1001 kkkk kkkk	39kk	AND Literal and W	ANDLW	k	k & W → W	Z	
10 0kkk kkkk kkkk	2kkk	Call subroutine	CALL	k	PC + 1 → TOS, k → PC <10:0>, PCLATH <4:3> → PC <12:11>;	None	
00 0000 0110 0100	0064	Clear Watchdog timer	CLRWDT	-	0 → WDT (and prescaler, if assigned)	TO, PD	
10 1kkk kkkk kkkk	2kkk	Go to address	GOTO	k	k → PC <10:0>, PCLATH <4:3> → PC <12:11>;	None	
11 1000 kkkk kkkk	38kk	Incl. OR Literal and W	IORLW	k	k v W → W	Z	
11 00XX kkkk kkkk	30kk	Move Literal to W	MOVLW	k	k → W	None	
00 0000 0000 1001	0009	Return from interrupt	RETFIE	-	TOS → PC, '1' → GIE	None	
11 01XX kkkk kkkk	34kk	Return,place Literal in W	RETLW	k	k → W, TOS → PC	None	
00 0000 0000 1000	0008	Return from subroutine	RETURN	-	TOS → PC	None	
00 0000 0110 0011	0063	Go into standby mode	SLEEP	-	0 → WDT, stop oscillator	TO, PD	
11 110X kkkk kkkk	3Ckk	Substract W from literal	SUBLW	k	k - W → W	C,DC,Z	
11 1010 kkkk kkkk	3Akk	Excl. OR Literal and W	XORLW	k	k ⊕ W → W	Z	
00 0000 0110 0010	0062	Load OPTION register	OPTION	-	W → OPTION register	None	1
00 0000 0110 0fff	006f	Tristate port f	TRIS	f	W → I/O control register f	None	1

CONCLUSION

Arrivé à ce stade de l'ouvrage, vous savez tout ou presque des microcontrôleurs PIC et vous pouvez vous lancer dans le développement de vos premières applications. Cependant, comme il est parfois difficile de débuter sur une nouvelle famille de microcontrôleurs ou bien encore qu'il est particulièrement pénible de devoir à chaque fois « réinventer la roue », sachez que nous avons rédigé pour vous, dans cette même collection, un ouvrage consacré aux applications des microcontrôleurs PIC.

Vous pourrez y découvrir :

- la présentation de différents systèmes de développement, ainsi que toutes les indications pour vous procurer gratuitement le merveilleux outil MPLAB de Microchip ;
- de nombreuses solutions d'interfaçage des microcontrôleurs avec le monde extérieur, avec des exemples complets, c'est-à-dire abordant tout à la fois l'aspect matériel et logiciel ;
- la présentation de nombreux sous-programmes standards qui ont été choisis pour leur intérêt, mais aussi pour la difficulté que peut présenter leur réalisation en langage machine, comme c'est le cas pour des sous-programmes de calcul en virgule flottante par exemple ;
- la description de trois applications complètes, retenues parce qu'elles font appel à des solutions matérielles ou logicielles qui n'ont pas été traitées par ailleurs dans l'ouvrage ;
- enfin, une rapide présentation des Basic Stamp de Parallax. Ce sont des pseudo microcontrôleurs, basés sur différentes versions de PIC programmés avec un intepréteur Basic orienté microcontrôleurs et automatismes.

Muni de ces deux ouvrages, les microcontrôleurs PIC ne devraient plus pouvoir vous résister. N'oubliez pas non plus que, si la langue anglaise ne vous pose pas de problème et si une documentation vous fait défaut, vous pouvez accéder au site Internet de Microchip d'adresse *www.microchip.com*, qui est particulièrement complet et très bien réalisé.

ANNEXE

CÉDÉROM MICROCHIP

Tout au long de cet ouvrage, nous vous avons fait découvrir les ressources internes des microcontrôleurs des familles 12Cxxx et 16Cxxx mais nous avons volontairement passé sous silence les paramètres électriques exacts des circuits, ainsi que la présentation de leurs brochages respectifs. Cette décision délibérée est justifiée par le fait que ces informations sont propres à chaque circuit et que, vu le nombre de membres que comportent ces familles, plusieurs centaines de pages auraient été nécessaires rien que pour cela.

Nous avons donc décidé, en association avec Microchip, de vous offrir le double cédérom Microchip dans sa dernière édition à la date de mise sur le marché de cet ouvrage. Vous disposerez ainsi avec lui de toutes les fiches techniques, notes d'application et documentations diverses de tous les circuits commercialisés par Microchip, et donc des microcontrôleurs objets de cet ouvrage. En prime, ce cédérom contient également un certain nombre d'outils de développement parfaitement fonctionnels et totalement gratuits, tel le célèbre MPLAB présenté dans notre ouvrage *Microcontrôleurs PIC - Applications* (du même auteur chez le même éditeur), qui vous permet de développer vos propres applications avec un maximum de confort et d'efficacité.

Comme c'est la coutume depuis déjà quelques années, ce cédérom est en fait la copie du site Internet de Microchip et il se lit donc avec le navigateur Internet de votre choix (Internet Explorer ou Nestcape Communicator). Son utilisation sur votre PC se déroule donc de la façon suivante :

– Si votre lecteur de cédérom est programmé pour ce que l'on appelle la notification d'insertion automatique, c'est-à-dire qu'il exécute automatiquement le programme de démarrage de tout cédérom que vous lui donnez à lire, placez le cédérom n° 1 dans le lecteur et attendez. Lorsque l'écran de démarrage s'affiche, choisissez « Launch Browser ». Ceci lance votre navigateur Internet par défaut et ouvre la première page du site Internet de Microchip sur le cédérom.

– Dans le cas contraire, insérez le cédérom n° 1 dans le lecteur, ouvrez l'explorateur de Windows et cliquez sur le fichier « index.htm ». Ceci lance votre navi-

gateur Internet par défaut et ouvre la première page du site Internet de Microchip sur le cédérom comme ci-dessus.

Vous pouvez alors parcourir le cédérom en cliquant sur l'information désirée, « PICmicro MCUs » dans le cas des microcontrôleurs qui nous intéressent par exemple.

Si vous ne possédez aucun navigateur Internet, le cédérom n° 2 vous propose Internet Explorer ainsi que Netscape Communicator dans leurs versions respectives les plus récentes. Vous pourrez ainsi installer celui de votre choix. Vous trouverez par ailleurs sur ce cédérom n° 2 le programme Winzip qui vous permettra de décompresser certains fichiers « zippés » présents dans diverses notes d'applications. Vous y découvrirez aussi Acrobat Reader d'Adobe qui vous permettra de lire les fichiers au format pdf, format qui est aujourd'hui un standard de l'échange de documentation électronique. Enfin, c'est également sur ce cédérom n° 2 que se trouvent les outils de développement, dont le fameux MPLAB évoqué ci-dessus.

Et comme Microchip est une société très dynamique ce qui, revers de la médaille, fait vieillir très vite ses documentations, rappelons que si vous ne trouvez pas la fiche technique du dernier circuit sorti sur ces cédérom, vous y aurez accès sur le site Internet de Microchip dont l'adresse est tout simplement :

www.microchip.com.

INDEX

12Cxxx, 5, 7
16C5x, 6, 8
16Cxxx, 6, 9-10

A

abonné, 119-120
acquittement, 120
ADCON0, 149-152, 155, 158
ADCON1, 75, 149, 152-153, 158-159
ADRES, 149
ADRESH, 157-158, 160
ADRESL, 157-158, 160
adressage, 183
  de type bit, 184
  direct, 183
  immédiat, 183
  indirect, 184
  literal, 185, 188
afficheur LCD, 113, 163-164, 170, 176
alimentation, 13
ALU, 41
arbitrage, 119, 124
architecture
  de Harvard, 3-4
  de Von Neumann, 3-4
  interne, 39-40
  RISC, 4
asynchrone, 132-133
  full duplex, 131

B

baisse de la tension d'alimentation, 32
Basic Stamp, 203
bit
  de configuration, 20
  de protection, 12, 24
BOR, 26, 32, 35-36

C

capture, 105, 107-108
CCP, 101, 105, 107, 156
CCPRx, 105, 107-108
CCPRxH, 105
CCPRxL, 105
CCPx, 107-108
CCPxCON, 105, 107, 110-111
CCPxH, 109
CCPxL, 109-110
chronogramme d'un reset, 35-36
chute de tension d'alimentation, 26, 35
CLRWDT, 70
CMCON, 75, 146-147
comparaison, 105, 108-109
comparateur, 141, 143, 146
  analogique, 144-146
  de tension, 143
compatibilité TTL, 13
convertisseur analogique/digital
  8 bits, 149-150, 153-155
  10 bits, 157, 159-161

D-E

débordement du *timer* chien de garde, 26, 71
échantillonnage-blocage, 149
échantillonneur bloqueur, 154
EEADR, 64-65
EECON1, 63-65
EECON2, 63-65
EEDATA, 64
erreur de parité mémoire, 26
esclave, 119, 122

F

famille
  12Cxxx, 5, 7
  16C5x, 8
  16Cxxx, 6, 9-10
flash, 12
fréquence de trame, 171
FSR, 53, 184
full duplex, 131

G-H

générateur de tension, 176
half duplex, 131, 134
Harvard, 3, 39
horloge, 19, 21-22, 168
  à quartz, 19, 22
  de conversion, 152
  RC externe, 19
  RC interne, 20, 23

I

I2C, 113, 118-122, 125-126
ICSP, 179
INDF, 50, 184
instruction, 185, 188
INTCON, 57-58, 80, 95
interaction CCP, 106
interface
  de communication série, 131
  LCD, 173
interruption, 56, 72

J-L

jeu d'instructions, 185, 188
latch-up, 148
LCD, 77, 163
LCDCON, 163-166, 173, 175
LCDD, 163-164, 168-169, 175
LCDPS, 163, 166-167, 170, 174
LCDSE, 75, 159, 167, 174

M

maître, 119, 122
mémoire, 41
  de données, 48-49
  de programme, 11, 52
  EEPROM de données, 63
  flash, 12
  OTPROM, 11
  PROM, 12
  ROM programmable par masque, 12
Microwire, 113
mise sous tension, 26, 31, 66
mode
  d'adressage, 183
  sommeil, 13, 70-71
modulation de largeur d'impulsions, 105, 109
MPLAB, 203
multiplexage, 168
  1/2, 173
  1/3, 174
  1/4, 175

N-O

niveaux TTL, 76
OPTION, 54, 68, 79, 85, 93-94, 187
OSCCAL, 24-25
OTPROM, 11

P

pagination
  de la mémoire de données, 47-48
  de la mémoire de programme, 42-43, 45-46, 53
patte, 14-17
PCALTH, 42
PCH, 43-44, 46
PCL, 43-44, 50
PCLATH, 43-44, 46, 53, 187
PCON, 37, 53
période du signal, 109
PIEx, 57, 59, 102
pile, 57-58
  LIFO, 58, 133
PIRx, 57, 61, 89
pompe de charge, 171-172
POR, 26
port
  A, 76-77
  B, 78-79
  C, 80-81
  D, 81-82
  D, 89
  E, 82-83
  F, 83-84
  G, 83-84
  GPIO, 84-85
  LCD, 84
  parallèle, 73, 147
  parallèle esclave, 86
  série synchrone, 113
  SPI, 114
PORTX, 73
PR2, 101
programmation en circuit, 178-180
PROM, 12
PSP, 75, 83, 86-87, 89
PWM, 105, 109-111

Q-R

quartz, 19, 22
read-modify-write, 74
rapport cyclique, 109
RCREG, 133
RCSTA, 134-137
référence de tension, 149
registre
  de calibration, 23
  interne, 48
  OSCCAL, 24
relatif, 184
reset, 26-30, 32-33, 67, 72-73
  automatique, 31-32
  externe, 31, 71
  manuel, 31
résonateur céramique, 22
réveil, 71-72
RISC, 4, 185
RTSS, 91

S

SCI, 131-135, 139
SCK, 114, 117
SCL, 120-123
SDA, 120-123
SDI, 114, 116
SDO, 114, 116
SLEEP, 67, 70-72, 80, 85, 95, 101, 147, 149, 156, 161, 164, 168, 175-177
SPBRG, 132, 134, 138-139
SPI, 113, 115, 117
$\overline{SS}$, 115, 117
SSI, 116
SSP, 113, 116, 118, 126-127
SSPADD, 125-126
SSPBUF, 115, 125-126
SSPCON, 125, 129
SSPSR, 115, 125

SSPSTAT, 125, 127
STATUS, 38, 47, 51-52, 57
synchrone, 135
half duplex, 134

T

T1CON, 99-101
T2CON, 103-104
tension
d'alimentation, 13
de référence, 141-143
timer
0, 91-92
1, 99, 106-108, 163, 170
2, 101-102, 106, 110
chien de garde, 67-72
TMR1, 100
TMR1H, 99
TMR1L, 99
TMR2, 101-103
TRIS, 74, 187
TRISA, 77
TRISB, 79
TRISC, 80-81
TRISD, 82, 87
TRISE, 83, 89
TRISGP, 84-86
TRISX, 73-74, 108, 116
TXREG, 131-132
TXSTA, 132, 135-136, 138

U

unité arithmétique et logique, 41
USART, 131

V

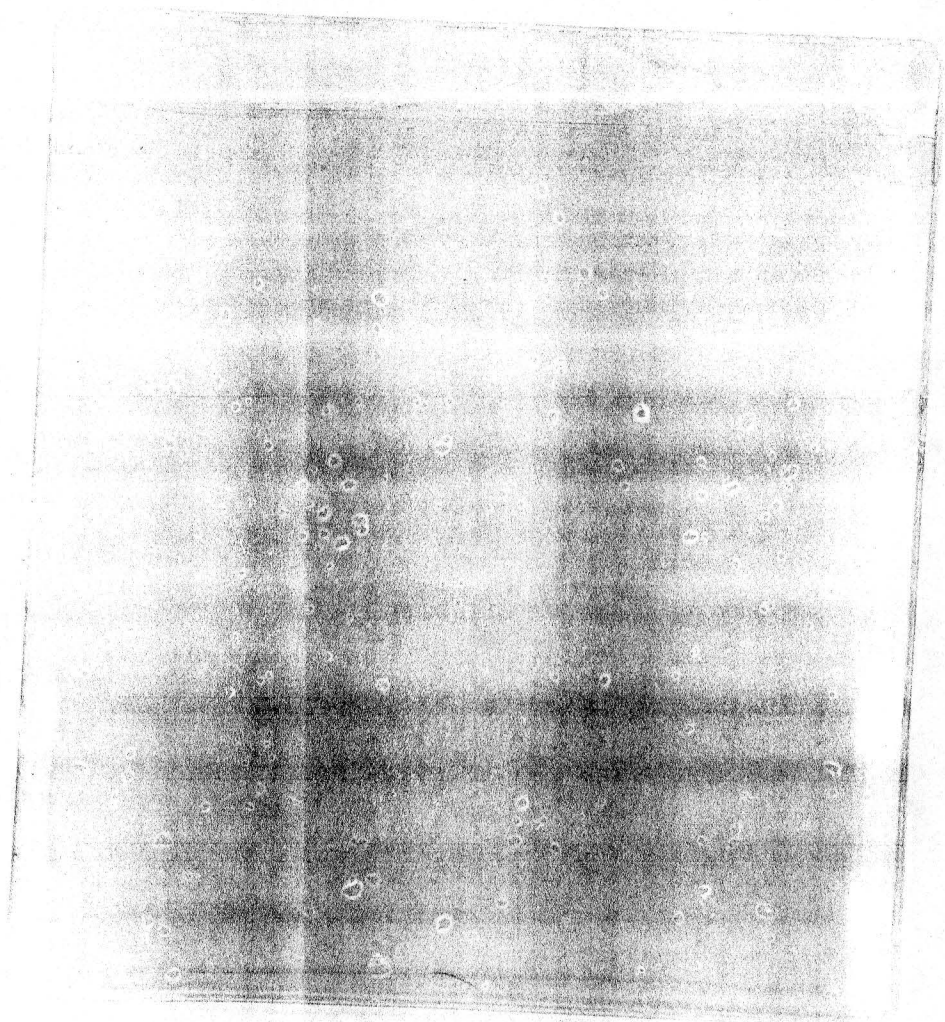
vecteur d'interruption, 57
verrouillage, 148
vitesse de la SCI, 138-139
Von Neumann, 3, 39
VRCON, 141-143

W

W, 41, 184
WDT, 67-69

045266-(I)-(2,5)-OSB 80°-ALL

STEDI, 1, boulevard Ney, 75018 Paris - Tél. 01.40.38.65.40
Dépôt légal. Imprimeur, n° 6643

Dépôt légal : octobre 2000

Imprimé en France